DOCUMENT RESUME

ED 065 144 LI 003 765

AUTHOR	Weiman, Carl F. R.; Rothstein, Jerome
TITLE	Pattern Recognition by Retina-Like Devices.
INSTITUTION	Ohio State Univ., Columbus. Computer and Information Science Research Center.
SPONS AGENCY	National Science Foundation, Washington, D.C. Office of Science Information Services.
REPORT NO	OSU-CISRC-TR-72-8
PUB DATE	Jul 72
NOTE	165p.;(28 References)
EDRS PRICE	MF-$0.65 HC-$6.58
DESCRIPTORS	*Algorithms; Computers; *Computer Science; Doctoral Theses; *Pattern Recognition
IDENTIFIERS	*Farey Series

ABSTRACT

This study has investigated some pattern recognition capabilities of devices consisting of arrays of cooperating elements acting in parallel. The problem of recognizing straight lines in general position on the quadratic lattice has been completely solved by applying parallel acting algorithms to a special code for lines on the lattice. The relation of the code to Farey series and continued fractions and the effects on the code of a line when the line is subjected to affine transformations were studied in detail. Algorithms for reducing straight line codes to a standard form were developed and made the basis of a line recognition process. Cellular automata were designed tc carry out line recognition. Other cellular automata were designed to recognize topological connectedness, detect boundaries and approximate curves by straight line segments. (Author)

ED 065144

U.S. DEPARTMENT OF HEALTH,
EDUCATION & WELFARE
OFFICE OF EDUCATION
THIS DOCUMENT HAS BEEN REPRODUCED EXACTLY AS RECEIVED FROM THE PERSON OR ORGANIZATION ORIGINATING IT. POINTS OF VIEW OR OPINIONS STATED DO NOT NECESSARILY REPRESENT OFFICIAL OFFICE OF EDUCATION POSITION OR POLICY.

(OSU-CISRC-TR-72-8)

PATTERN RECOGNITION BY RETINA-LIKE DEVICES

by

Carl F. R. Weiman and Jerome Rothstein

Work performed under

Grant No. 534.1, National Science Foundation

The Computer and Information Science Research Center
The Ohio State University
Columbus, Ohio 43210

July 1972

LI 003 765

ABSTRACT

This study has investigated some pattern recognition capabilities of devices consisting of arrays of cooperating elements acting in parallel. The problem of recognizing straight lines in general position on the quadratic lattice has been completely solved by applying parallel acting algorithms to a special code for lines on the lattice. The relation of the code to Farey series and continued fractions and the effects on the code of a line when the line is subjected to affine transformations were studied in detail. Algorithms for reducing straight line codes to a standard form were developed and made the basis of a line recognition process. Cellular automata were designed to carry out line recognition. Other cellular automata were designed to recognize topological connectedness, detect boundaries and approximate curves by straight line segments.

PREFACE

This work was done in partial fulfillment of the requirements for a doctor of philosophy degree in Computer and Information Science from The Ohio State University. It was supported in part by Grant No. GN-534.1 from the Office of Science Information Service, National Science Foundation, to the Computer and Information Science Research Center of The Ohio State University.

The Computer and Information Science Research Center of The Ohio State University is an interdisciplinary research organization which consists of the staff, graduate students, and faculty of many University departments and laboratories. This report is based on research accomplished in cooperation with the Department of Computer and Information Science.

The research was administered and monitored by The Ohio State University Research Foundation.

ACKNOWLEDGMENTS

I wish to acknowledge my debt to Professor Jerome Rothstein, my advisor for this dissertation. He is responsible for clearly formulating the problem studied, inventing the code, and discovering many of its properties. This material constitutes the bulk of the first half of the dissertation (the first four chapters). He also suggested the application of Farey's series and continued fractions to the problem. The resulting theorems on continued fractions and their relation to the code are largely my work.

The second half of the dissertation, containing the parallel computer designs incorporating the code properties, is primarily my work. Professor Rothstein provided valuable suggestions and no less valuable editing of my often unreadable text describing the operations of the parallel computers.

Many thanks to my reading committee, Professors Weed, Saltzer, and Ernst for their valuable suggestions, patience, and tolerance of exposure to very rough drafts given them much too close to deadlines.

In addition I wish to thank Dudley Fulton for writing a Fortran IV G simulation of a straight line recognizing parallel computer similar to that described in V.2. His work provided a test of the design that led to improvements. Conversations with him also clarified my thinking on the design of parallel computers.

I also wish to thank Professor Rothstein personally, not only for the already described debt concerning this dissertation, but also for skillfully introducing me to the many agonies and rare (but greater)

ecstasies that accompany research. A contributing factor in my motivation to do research has been exposure for several years to his courses and personal conversations, many of which though unrelated to this dissertation directly, inspired me indirectly to look at the world in a much more interesting way.

TABLE OF CONTENTS

		Page
ABSTRACT		ii
PREFACE		iii
ACKNOWLEDGMENTS		iv
LIST OF FIGURES		viii
Chapter		
I.	INTRODUCTION	1
II.	STRAIGHT LINE CODE	5
	Code Definition Generalization of Code to All Octants Elementary Code Properties Resolving Power of Code for Translation of Lines Slope Resolution on Finite Grids	
III.	AFFINE TRANSFORMATIONS AND STRAIGHT LINE CODES	27
	Unimodular Transformations Algorithms for Recognition of Straight Line Codes	
IV.	CODE TRANSFORMATIONS, CONTINUED FRACTIONS AND FAREY SERIES	40
	Code Reduction and Continued Fractions Translation of Lines and Cyclic Shifts in Code Cyclic Shifts and Code Concatenation Shift Polygons Lines as Operators on the Code Rotation of Lines and Farey Series in Terms of Continued Fractions	
V.	PARALLEL COMPUTERS WHICH RECOGNIZE AND GENERATE STRAIGHT LINES	81
	Structure and Notation of Parallel Computers Straight Line Recognizer Based on CCF-Reduction	

Page

Straight Line Recognizer Based on Shift Polygon
Straight Line Generator
Discussion of Parallel Computers

VI. PARALLEL COMPUTERS TO RECOGNIZE CONNECTIVITY AND OTHER PROPERTIES 119

Separation of Distinct Connected Regions
Determination of Boundaries of Connected Regions on the Grid
Straight Line Approximation of Curves
Aspects of Polygon Recognition

VII. CONCLUDING REMARKS. 147

APPENDIX: ORIGIN OF THE CODE (by J. Rothstein) 149

BIBLIOGRAPHY 152

LIST OF FIGURES

			Page
1.	II.1:	Code Definition and Octant Symmetry	7
2.	II.2:	Run and Bend Configurations and Compatible Octants	10
3.	II.3:	Lattice Geometry for Cyclic Shift in Code	16
4.	II.4:	Straight Line Code and Farey Series	24
5.	III.1:	Effect of Shearing Transformation on Code	30
6.	IV.1:	Code Reduction and Continued Fractions	44
7.	IV.2:	Geometric Interpretation of CF and CCF Convergents	49
8.	IV.3:	Lattice Geometry Related to Cyclic Shift in Code	55
9.	IV.4:	Relation of Shift Polygon to Code and Continued Fraction	59
10.	IV.5:	Function Relating Slopes with Fixed Denominators to Shift Polygons with Fixed Number of Vertices	62
11.	IV.6:	Transposition Table for Cyclic Shifts	65
12.	IV.7:	Operator Plane	70
13.	IV.8:	Geometric Representation of p/q and its Predecessors in all Farey Series	77
14.	V.1:	Organizational Hierarchy of Parallel Computers	83
15.	V.2:	Octant Compatibility Detector	88
16.	V.3:	Line Recognizer Based on CCF-Reduction	92
17.	V.4:	Straight Line Recognizer Based on Shift Polygon	99
18.	V.5:	Interconnections for Straight Line Generator	106

			Page
19.	V.6:	Parallel Computer that Determines and Stores Slope of Line to be Constructed	108
20.	V.7:	Parallel Computer that Constructs a Line, Given its Slope	112
21.	VI.1:	Parallel Computer that Recognizes Connected Regions	121
22.	VI.2:	Interconnections for Assigning Regions to CS's	123
23.	VI.3:	Parallel Computer to Recognize "Boundaries"	127
24.	VI.4:	Interconnections for Straight Line Approximation	130
25.	VI.5:	Parallel Computer that Holds C's not Currently Being Processed by Line Recognizer or Illegal String Truncator	131
26.	VI.6:	Straight Line Recognizer Used in the Approximation of Curves by Line Segments	133
27.	VI.7:	Parallel Computer that Truncates Illegal Substring	136
28.	VI.8:	Grid Cells Crossed by Lines that Meet at Obtuse Angles	142
29.	VI.9:	Interference of Excited Neighborhood Configurations Near Acute Vertices	144

CHAPTER I: INTRODUCTION

The purpose of this study was to gain insight into the pattern recognizing capabilities of devices consisting primarily of arrays of cooperating elements acting in parallel. A formal system of this kind was studied rigorously.

Arrays of simple devices are integrated into a system that can, by virtue of its structure, perform complex tasks far beyond the capabilities of the individual elements. An array of identical finite state automata is arranged on a quadratic lattice, one per lattice cell, with inputs and outputs connecting each automaton to its four nearest neighbors. Each automaton also has an external input. In addition, there are central synchronizing automata, each capable of receiving inputs from or sending outputs to large numbers of the array automata simultaneously. The array of automata is a member of the class of mathematical structures or systems variously called cellular automata, iterative circuit computers, tesselation automata, and parallel computers (see Yamada and Amoroso, 1969; Burks, 1970; Codd, 1968; VonNeumann, 1966, Gonzalez, 1963; Garner and Squire, 1963 and Holland, 1959). The function of the central synchronizer and the method by which array automata connect to it are departures from the preceding studies. The result is increased flexibility in the operations of the array automata.

The system chosen was inspired in part by visual retinas. The system shares certain properties with visual retinas. Both are two-dimensional arrays that process patterns by the simultaneous actions

and interactions of individual devices, and central control units are vital to operation of the arrays. In this sense, and virtually only in this sense, can this and similar devices be called retina-like. However, other loose analogies often crop up. For example, laterla inhibition is a kind of local neighborhood interaction, (see Ratliff, 1965). Also, the shift polygon recognizer of section V.3 uses what can be viewed as an analog of eye tremor.

The designs of some pattern recognition devices have been motivated by properties of retinas. One of the earliest and most thoroughly studied is the perceptron (see Rosenblatt, 1962; Block, 1962). It uses receptive fields and simple summation of effects from neighboring cells. Its failure to recognize simple topological properties and simple figures (e.g., polygons) without great complications has been extensively analyzed by Minsky and Papert (1969). Here the neighborhood functions can be completely general, permitting complicated discriminations of excited configurations of cell niehgborhoods over which the perceptron merely sums. Neural networks (see McCulloch, 1943; Caianiello, 1968) have many of the same limitations as perceptrons, particularly the use of threshold functions and fixed connections between components.

To study the pattern recognition capabilities of cellular automata, we chose a specific class of patterns, thoroughly studied its properties on the grid, and then designed programs to exploit those properties. To make the pattern class simple enough to yield transparent results and complex enough to be of interest, the class of straight lines was

chosen. Lines are the simplest non-trivial elements in Euclidean geometry. Points, in contrast, are too simple because their recognition involves excitation of a single detector rather than a pattern of excitation in a group of detectors. Straight lines are also the simplest functions in analytic geometry and the only ones expressible in linear form in planar cartesian coordinates. Furthermore, straight lines are central to geometrical optics and the mechanics of rigid bodies, two fields that loom large in the world of visual experience. Finally, since any pattern consisting of curves can be approximated to any desired accuracy by straight line segments, the door is open to generalization.

Once straight lines were chosen as the pattern class, a special code was constructed and used to represent the position of lines with respect to lattice cells. It has several uniquely useful properties. For example affine (line preserving) transformations of the line correspond to simple transformations on the code, such as deleting and interchanging digits. Since affine transformations are important in the projection of three-dimensional objects onto two dimensional surfaces (e.g., retinas) this property is valuable for pattern recognition. In particular, translation and rotation of patterns on the lattice are easily treated.

After the relation of the code to line geometry was understood, automata programs were designed to recognize lines. These programs and the structure of the automata executing them are invariant under translation and rotation of lines. Also, the programs operate largely

in parallel. That is, recognition is the result of a number of steps, each consisting of the simultaneous processing of local neighborhood information by all relevant cells.

The use of local neighborhoods of excitation as the basis for recognition superficially resembles the use of feature or masks in standard pattern recognition methods. These treat recognition as a problem of statistical decision (see for example Sebestyen, 1962; Fu, 1968; and Patrick, 1972). This is in sharp contrast to the algorithmic approach of this study, which is exact and not probabilistic.

The straight line recognition program was the first successful one. Other programs were constructed to recognize topological connectivity, detect boundaries, and recognize complex configurations of lines.

CHAPTER II: STRAIGHT LINE CODE

To design cellular automata algorithms for pattern recognition, a notation or code for describing configurations of excited lattice cells is needed. Properties of particular configurations then reduce to properties of the code which would be incorporated into the design of recognition algorithms. To be useful, such a code should be economical in that it carry only information of interest, and transformations of the pattern relevant to recognition should result in simple code transformations. Raster scan codes can be rejected on the basis of both criteria. First, they are not economical since they include information from all lattice cells, not just those excited by the pattern. Second, code digits corresponding to cells excited by a pattern may be scattered among digits corresponding to cells that are not; thus the changes in such a code resulting from geometric transformations of the pattern (e.g., translation, rotation) may be quite complicated.

II.1 Code Definition

A code in which each digit corresponds to a single lattice cell excited by a line can be defined by observing that the line must intersect exactly two sides of each cell it crosses and the symmetry of the location of these sides can only be of two types. The two sides crossed must be either parallel or perpendicular to each other. If the line goes through a lattice point, that point can be assigned to two perpendicular sides of a lattice cell in a manner consistent with the

above observation. In the code to be defined below a 0 corresponds to a cell in which parallel sides are crossed by the line and a 1 corresponds to a pair of adjacent cells in which perpendicular sides are crossed. In the latter case a single digit corresponds to a pair of cells because such cells always occur in pairs. The more precise definition given below will be used to derive straight line code properties to be used in the design of recognition algorithms. Some of these properties will also be used to show that the restriction in the definition that lines be rays in octant I* starting at the origin can be relaxed.

Definition II.1: Given a ray in octant I starting at the origin, generate its code by writing 0 for each lattice cell it crosses through parallel sides and 1 for each cell it crosses through perpendicular sides, omitting the cell immediately following in the latter case. Lattice points are considered as the corners of cells immediately above and to the left. (See Figure II.1.)

The convention that a lattice point be considered as lying on the side of a particular lattice cell is adopted so that each cell crossed by a line have exactly two nearest neighbor cells also crossed by that line (with the cell adjacent to the origin the sole exception). Of the four cells bordering a lattice point crossed by a line, two are already crossed by the line so the choice of cell membership for the point must be made between the remaining two cells. This choice is arbitrary; the upper left cell was chosen in the above definition so that the origin would be considered as belonging to the cell above and to its left.

---

* The octants of the plane are defined as the eight wedges separated by the lines x=0, y=0, x=y, and x=-y. They are numbered in counter-clockwise order with roman numerals, octant I being the wedge immediately above the positive x-axis.

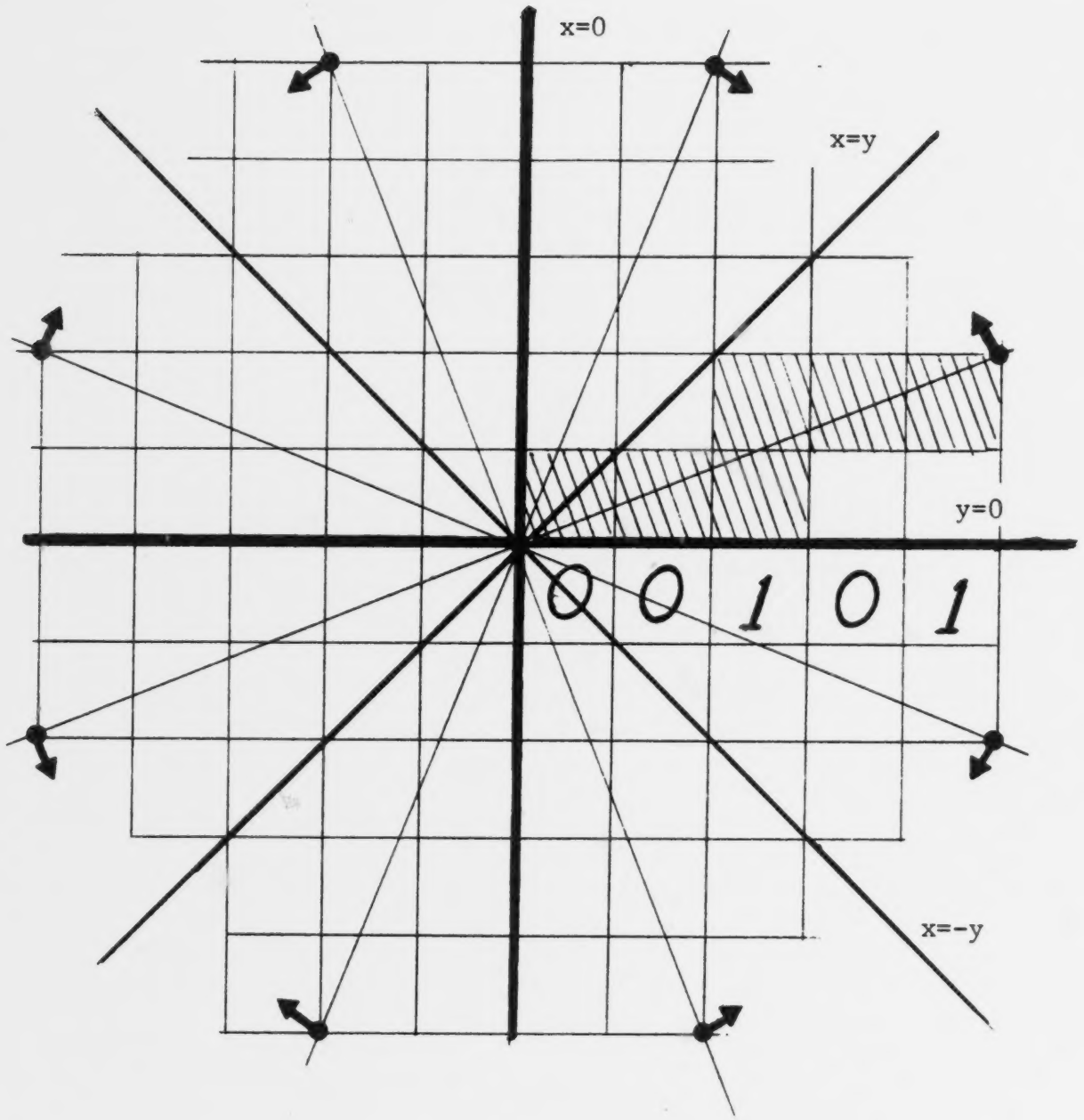


Figure II.1: Code Definition and Octant Symmetry

Since this cell shares a vertical side with the first cell crossed by the line, and a line of slope $m < 1$ must go through the vertical side on the right of the first cell crossed by the line, parallel (vertical) sides of the first cell are crossed by the line and the first digit of any line code is 0. Were the cell below and to the right of a lattice point considered as containing that point instead, the first digit of any line code would be 1.

II.2 Generalization of Code to All Octants

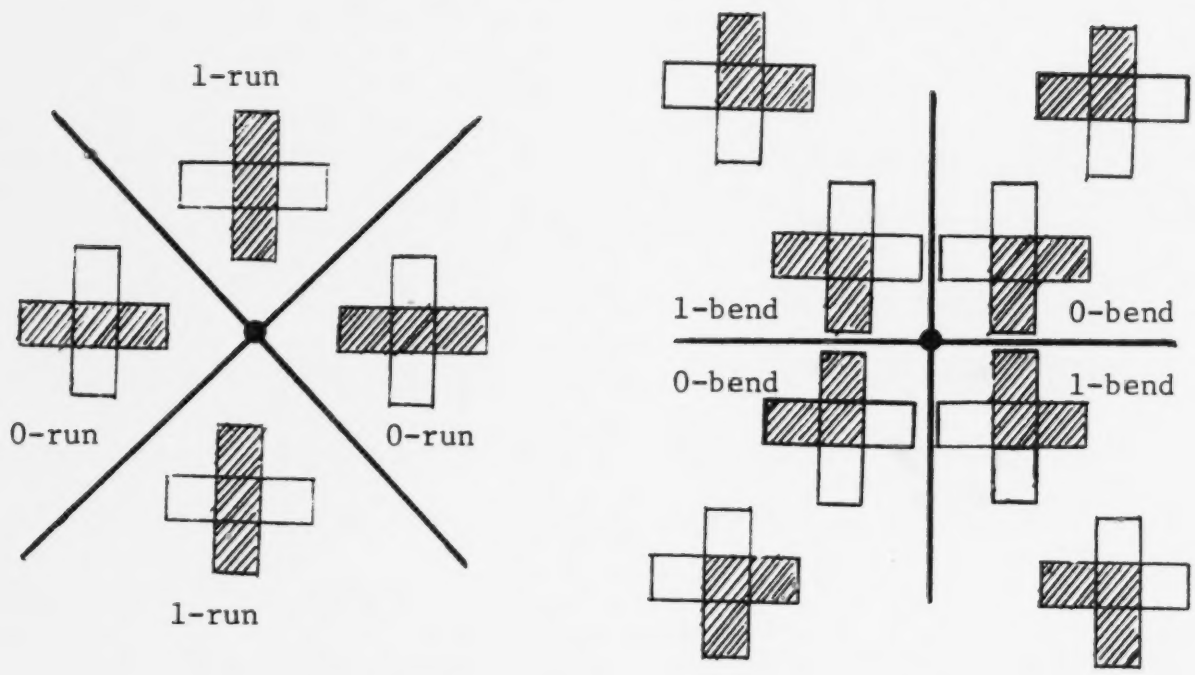
Definition II.1 will yield the code of a line in any octant if the convention for lattice points crossed by the line is appropriately changed. Note that reflecting the lattice about any of the lines $x=0$, $y=0$, $x=y$, or $x=-y$ (x- and y-axes and two main diagonals) leaves it unchanged. Equivalently, reflecting any line about those axes leaves its code unchanged since the symmetry of sides of cells it crosses is unchanged. To be consistent with the convention that the first digit of a code be 0, the location of cells to which lattice points belong must be chosen so that the reflection that brings the line into octant I also brings the cell location into that of definition II.1. The resulting convention for the various octants is shown by the small arrows in Figure II.1 pointing from the lattice points to the cell to which they belong.

The eightfold ambiguity in the interpretation of a particular code corresponding to the possible octants in which its line lies can be resolved by adding three bits to the code. These correspond to the

orientation of neighborhood configurations of cells corresponding to 0's and 1's and the direction in which the line is traced. Since a code 0 corresponds to a cell with parallel sides crossed by a line, and those sides are shared by nearest neighbors on opposite sides of the cell in question, these nearest neighbors must also be crossed by the line. An excited cell with such a configuration of excited nearest neighbors will be called a run. If the excited neighbors are to the right and left the cell is called a 0-run and if they are above and below it is called a 1-run. These are the only kinds of runs possible and the two kinds can only arise in configurations of cells crossed by lines in the quadrant pairs shown in Figure II.2. Similarly, a code 1 corresponds to a pair of cells, each of which has perpendicular sides crossed by a line. Thus, each excited cell in the pair has two excited nearest neighbors on adjacent sides. Although there are four possible such configurations, which will be called bends, they always occur in pairs which can only arise in configurations of cells crossed by lines in the quadrant pairs shown in Figure II.2. Thus, as with the runs, there are two mutually exclusive bend types labelled 0 and 1 according to the quadrant compatibility shown in Figure II.2. Hence, the code of a line may be interpreted as a sequence of bend pairs and runs representing the lattice's approximation of the line.

Specifying the type of run or type of bend pair reduces the ambiguity of line orientation in interpreting a code by narrowing the possibility to four octants (a pair of opposite quadrants). From Figure II.2 it is obvious that specifying both run and bend types is equivalent to

Shaded Squares are Excited Cells



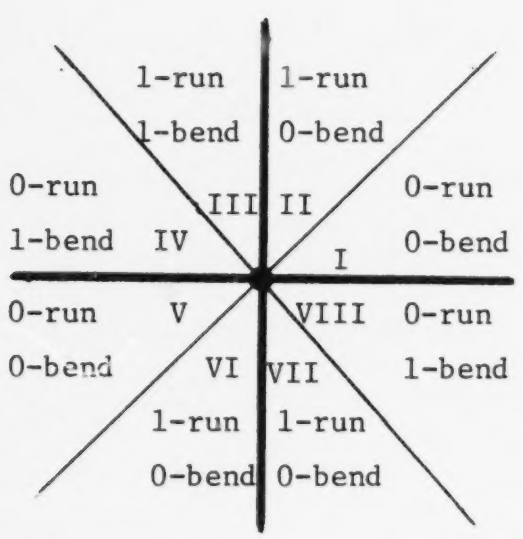


Figure II.2: Run and Bend Configurations and Compatible Octants

taking the intersection of two quadrant pairs, thereby narrowing the possible line orientation to a pair of octants 180^{o} apart in the lattice. This remaining ambiguity is that of the direction in which a line is traced. Adopting the convention that the positive direction is that which yields a code that starts with 0, this final ambiguity may be resolved by specifying the cell that a lattice point crossed by the line belongs to. From the discussion in the paragraph following definition II.1 recall that there are two possible choices for this convention, one yielding a code beginning with 0 for the line traced from the origin and the other yielding a code beginning with 1. The latter choice yields the same code as the former choice if the line traced is rotated 180^{o}, i.e., if it is traced in the opposite direction from the origin.

A line in any octant may be brought into octant I by appropriate lattice preserving reflections about the lines x=0, y=0, x=y, x=-y without changing its code. Thus, the code of any line in any octant is the same as the code of some line in octant I and no generality is lost in restricting the study of code properties to lines in octant I. This restriction is therefore adopted to simplify the proofs of the following theorems about elementary code properties. A finite subset of the lattice is called a grid in the following discussion.

II.3 Elementary Code Properties

Theorem II.1: The codes of all lines on an n x n grid are n digits long.

Proof: The lattice parallels divide the grid into n vertical columns. In a column in which only one cell is crossed by the

line this cell must be a run, yielding a code 0, since the line must enter the left side of the cell and exit the right in traversing the column. In a column containing a bend, the line must enter the column through the left side of a cell and then exit that cell through its upper side. (If it exited that cell through the right side the cell would be a run and the only excited cell in the column.) The second excited cell in this column must be entered through its lower side since it shares that side with the first excited cell. The line must then exit the column through the right side of the second excited cell. (If it exited through the upper side,the second cell would be a 1-run implying a slope constraint incompatible with octant I.) The first excited cell in this case yields a code 1 and the second is ignored, and a single digit per column results, completing the proof.

Theorem II.1 shows that code length is the same for lines of all slopes on a fixed grid and that this length is a natural fit to the grid size. Also, the proof shows that bends always occur in pairs and thus justifies the convention in definition II.1 that code 1's correspond to bend pairs.

Not all binary sequences are the codes of straight lines. An intuitive constraint on digit distribution is that 1's and 0's be distributed as homogeneously as possible. More precisely, since lattice parallels are uniformly spaced,a straight line is cut into equal segments by its intersections with either the horizontal or vertical lattice

parallels. As the intersections of the line with vertical parallels determine code digit columns and the intersections with horizontal parallels occur in columns corresponding to code 1's, the number of adjacent 0's between 1's anywhere in the code of any particular line should be nearly the same. Defining a segment of code of length k as a substring not preceeded by a 0 and consisting of k-1 consecutive 0's and the 1 following, the above intuition may be formalized as:

Theorem II.2: The lengths of any two segments in the code of a line cannot differ by more than one.

Proof: A segment of length k corresponds to a chain of k-1 adjacent runs terminated on both ends by bends. Since the line must cross horizontal sides of both these bends the limits on slope m imposed by this configuration of excited cells is:

a) $1/(k+1) < m < 1/(k-1)$

and thus for a segment of length k+h

b) $1/(k+h+1) < m < 1/(k+h-1)$

These two requirements are compatible for a fixed m only if h=0, 1 or h=0, -1.

Expressions a) and b) in the proof above show that the ratio of 1's to digits in a segment (i.e., the reciprocal of segment length) is nearly equal to the slope. The theorem below shows that the ratio of 1's to digits in certain longer substrings of the code of a line is equal to the slope.

Theorem II.3: The code of a line of rational slope p/q through the origin, where p and q are mutually prime positive integers and p<q, is periodic with period q and p 1's per period. Conversely, the slope of a line corresponding to a periodic code is rational.

Proof: The only lattice points crossed by the line have coordinates (kq,kp), k all integers. Thus, the figure consisting of the line and lattice cells it crosses from the origin to (q,p) is congruent to the similarly defined figure between any two lattice points crossed consecutively by the line. Hence the code corresponding to the q columns covered by each of these figures is identical and from the proof of Theorem II.4 consists of q digits.

Also from the proof of Theorem II.1, code 1's correspond to columns in which the line crosses lattice horizontals. Since there are p such crossings between (kq,kp) and ((k+1)q, (k+1)p) there are p 1's in every period of the code.

Converse: Construct the chain of runs and bends corresponding to a periodic code with q digits per period, p of which are 1's. Corresponding to any bend is a bend q columns to the right by periodicity of the code. A horizontal side of each of these bends must be crossed by the line, thereby constraining the slope to $p/(q+1) < m < p/(q-1)$. Similarly, for the line to cross corresponding bends k periods apart, the slope is bounded by $kp/(kq+1) < m < kp/(kq-1)$. Taking the limit as $k \to \infty$ yields $m=p/q$.

Theorem II.3 implies that a straight line whose code is aperiodic has irrational slope and conversely. Thus periodicity of the code of a line is equivalent to rationality of its slope and aperiodicity is equivalent to irrationality of its slope. In the periodic case the number of 1's per period is the numerator of the slope and the number

of digits per period is the denominator. In the aperiodic case the density of 1's (number of 1's divided by number of digits in a connected substring) approximates the slope with increasing accuracy as the length of the substring is increased.

The preceding theorems show that the code takes into account both the slopes of lines and the sizes of the grids they are projected on in a simple way. The theorems in the following sections (II.4 and II.5) show the effects of translation and rotation of a line on its code.

II.4 Resolving Power of Code for Translation of Lines

Theorem II.4: Vertical translation of a line of rational slope p/q results in a succession of cyclic shifts of fixed magnitude in its code. Reversing the direction of translation reverses the direction of the shift. The original code recurs after one unit vertical displacement.

Proof: (Refer to Figure II.3) Let the equation of the line be $y=(p/q)x+y_0$. Then the value of the y-intercept y_0 is the magnitude of the vertical displacement of the line from its original position through the origin. As y_0 is increased from its original value of zero the line moves off the lattice points (kq,kp), k all integers, but the code remains unchanged until new lattice points lie on the line because until that happens the sides of lattice cells crossed by the line are the same as when the line went through the origin. When a new lattice point (q', p') is crossed by the line due to its upward translation, the 1 in the code column corresponding to the lattice column to the right of that point and the 0 left of it are transposed.

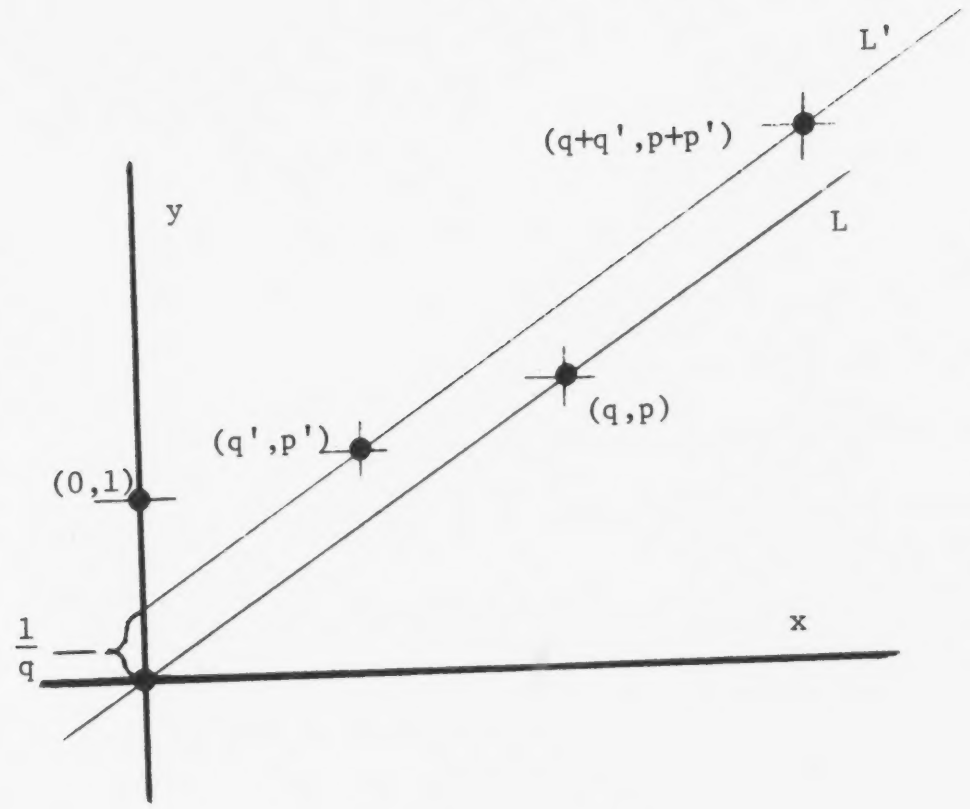


Figure II.3: Lattice Geometry for Cyclic Shift in Code

(That is, the line crossed perpendicular sides of the right cell and parallel sides of the left cell prior to crossing the point (q', p') and vice versa afterwards.) This transposition must occur in the same columns of every period of the code due to congruence of the geometry of the line on the lattice between lattice points lying on the line. It can only occur once in each code period since closer spacing would imply that the denominator of the slope of the line is less than q.

Clearly the geometry of the line $y=(p/q)x+y_0$ (where y_0 is the value for which (q', p') lies on the line) on the lattice between (q',p') and (q+q', p+p') is congruent to the geometry of the line y=(p/q)x on the lattice between (0,0) and (q,p). Thus, the code of the former line is the same as the code of the latter shifted q' digits to the right. Since the code is periodic, this is a right cyclic shift of q' digits in each period of the code. It also follows from this congruence that continued upward translation results in repetition of these cyclic shifts of magnitude q' and downward translation results in repeated left cyclic shifts of magnitude q'.

The magnitude of vertical displacement necessary to casue a single cyclic shift and hence the number of cyclic shifts to regain the original code are found in the following:

Lemma: For a line of slope p/q, vertical displacement by 1/q units results in a cyclic shift of the code by q' digits to the right and only multiples of q such shifts yield the original code of the line through the origin.

Proof: It is a property of the quadratic lattice that the minimun area of a parallelogram whose vertices are lattice points is one

unit squared (immediate consequences of theorems in Hardy & Wright, (1965), pp. 27-29). Since (q',p') is the first point the line crosses during upward translation, the points (0,0), (q,p), (q',p') (q'+q, p'+p) are the vertices of such a minimal parallelogram whose area is $q \cdot y_0 = 1$ implying $y_0 = 1/q$.

A succession of q such translations results in one unit vertical displacement of the line and hence the same sequence of cell crossings as the line through the origin and hence the original code. If q' and q had any common factors the original code might occur for multiples of the cyclic permutation other than q. In that case, the area of the minimal parallelogram could be written:

$$p'q-q'p=p'ak-bkp=k(ap'-bp) = 1$$

This implies that the common factor, k, of q and q', is 1.

There is only one degree of freedom of motion for an infinite line of fixed slope in the plane. This was expressed in Theorem II.4 as a variation in y-intercept in the equation of the line. Had the x-intercept been chosen to express this variation, the equation of the line could have been written

$$y=(p/q)x+y_0=(p/q)(x+(q/p)y_0)$$

where $-(q/p)y_0$ is the x-intercept. In that case, the horizontal translation necessary for a cyclic shift by q' digits to the right would be -1/p. Whatever convention is used to derive the properties of the code under translation of the line, the conclusion is that on either

side of a line of slope p/q going through lattice points there lies a strip of width $1/(p^2+q^2)^{-(1/2)}$ containing no lattice points and any translation of a line within each of these strips results in no change in its code. This expresses the finite resolving power of the code to distinguish the positions of lines of the same slope on the lattice. Note that this resolving power is generally much better than the resolving power of the grid in locating points ($1/(p^2+q^2)^{-1/2}$ for lines vs. 1 for points). This improvement is due to the fact that the line imposes a relation on the positions of p+q excited cells while the point excites only a single cell.

Theorem II.5: Of all possible cyclic permutations of the code of a line of rational slope, the code of the line through the origin has the minimum value when interpreted as a binary number.

Proof: Only upward displacements of the line by less than one unit need be considered to get all cyclic permutations of the code since unit vertical displacement yields the same code as no displacement and hence the code for $y=(p/q)x+y_0$ is the same as the code for $y=(p/q)x-(1-y_0)$. For upward displacement of a line, 1's shift to the left as a result of the transposition described in the proof of Theorem II.4 thereby increasing the value of the code interpreted as a binary number. The only exception occurs when a 1 in the first column shifts to the left and hence into the qth column. This only occurs when a line moves upward into a position exactly one unit above its original position through the origin, i.e., when the code is shifted back into its

original form with minimum binary value.

Displacing a line through the origin downward by less than 1/q units transposes the 0 at the beginning of each period of the code and the 1 at the end, leaving all other digits unchanged. This new code is the same as the code of a line through the origin based on the convention that a lattice point belongs to the cell below and to its right rather than above and to its left as in definition II.1. Thus the choice of convention for cell membership of lattice points is equivalent to choosing whether to regard lines through lattice points as lying slightly above or slightly below the points in question.

Recall from the discussion following definition II.1 that tracing a line in one direction yields the same code as tracing it in the opposite direction using the alternate convention of lattice point assignment to lattice cells. Thus, tracing a line from the origin to (q,p) using the standard convention yields the same code as tracing it from (q,p) to the origin using the alternate convention. Since from the above paragraph the only code change resulting from changing this convention is the interchanging of the 0 and 1 at the ends of a period, the intervening digits must read the same in forward and reverse order. In summary:

Theorem II.6: Each period of the code of a line of rational slope, excluding the first and last digit, is symmetrical, i.e., it reads the same in forward and reverse order. Either the first digit of a period is 0 and the last 1 or vice versa, depending on the convention chosen for lattice point assignment to lattice cells.

Consider the infinite binary sequence corresponding to the code of a line of irrational slope m through the origin ($y=mx+y_0$, $y_0 = 0$). Since the line crosses no lattice points other than the origin, adopting either the standard or alternate convention for lattice point membership in a cell affects only the first digit of this sequence. As the line is translated vertically by an arbitrarily small amount y_0, digits will shift in infinitely many places since the line will sweep over the lattice points representing the infinitely many rational approximations p/q to m such that $|p - qm| < |y_0|$. If y_0 is irrational, no lattice points lie on the line and if y_0 is rational p/q, the point (q,p) is the only lattice point lying on the line and the code is shifted q digits to the right. These and other properties of codes of lines of irrational slope will be important in the context of computing rational approximations of irrational numbers, particularly quadratic irrationals. For lines on finite grids, however, all codes are finite and in the following section (II.5) it will be shown that these finite codes represent the lines whose slopes are the best rational approximations of the irrational slopes on a given grid.

II.5 Slope Resolution on Finite Grids

On an nxn grid the code of a line of slope 0 through the origin consists of n consecutive 0's. As the line is rotated counterclockwise its code remains unchanged until its slope reaches 1/n. Then the lattice point (n,1) lies on the line and its code becomes (n-1) consecutive 0's followed by a 1. With continued rotation, the code

remains unchanged until the line sweeps across new lattice points. Each time this occurs the 1 in the code column corresponding to the lattice column to the right of such a point and the 0 left of it are transposed in exactly the same manner as described in the proof of Theorem II.4. Hence:

Theorem II.7: The value of the code of a line through the origin interpreted as a binary number increases monotonically with slope.

When the slope of the line reaches 1 (45^{o} rotation) its code is n consecutive 1's. Thus 0's correspond to a horizontal path and 1's to a diagonal path on the gird. With increasing slope the fraction of digits which are 1's increases monotonically.

Obviously a line will sweep across all lattice points in octant I as its slope is increased from 0 to 1. Whenever the line sweeps across a lattice point whose coordinates are not mutually prime, say (kq,kp) it also sweeps across (q,p). Therefore its code is periodic with q digits per period, p of which are 1's. Thus, rotating a line through the origin on an nxn grid yields the sequence of codes of lines with rational slopes p/q such that p and q are all integers satisfying:

$$0 \leq p \leq q \leq n$$

and the slopes are arranged in increasing order. This sequence of fractions is well known in mathematics as the Farey series of order n, written $\mathcal{F}_n$. (See Hardy & Wright, (1965), p. 23 and following for definition and properties of $\mathcal{F}_n$). Hence:

Theorem II.8: There is a 1-to-1 correspondence between the Farey series of order n and the codes of lines through the origin in octant I on an nxn grid.

Thus, theorems about Farey series correspond to properties of the code.

In the preceding discussion it was shown that the infinite class of lines through the origin with slopes between 0 and 1 is partitioned into a finite number of classes by the finite number of n-digit line codes. This expresses the finite resolving power of the grid to distinguish slopes of lines on an nxn grid. Each class characterized by a single code consists of all lines in a wedge whose apex is the origin and whose interior contains no lattice points (if there were interior points, the code would change as the line swept across them). The code of any line in such a wedge is the code of the line of rational slope through a lattice point on the lower boundary of the wedge. A single period of such a code is the shortest substring starting at the left that can be repeated to yield the entire code. The upper boundary of each wedge is a line whose slope is the successor in $\mathcal{F}_n$ of the slope of the lower boundary. Figure II.4 illustrates the relation between the wedges, 4-digit codes on a 4x4 grid, and $\mathcal{F}_4$. The overbar (—) on codes indicates one complete period.

Note that extending the nxn grid introduces new lattice points within the extensions of some of the wedges of the original grid. Thus, the partition of slopes by wedges corresponding to distinct codes is refined. However, since these new lattice points lie outside the original grid, the new codes resulting from their introduction are derived by simply appending digits to the original codes. That is, the new codes differ from the original codes only after the nth place. The rate at which these refinements are introduced as the grid is extended

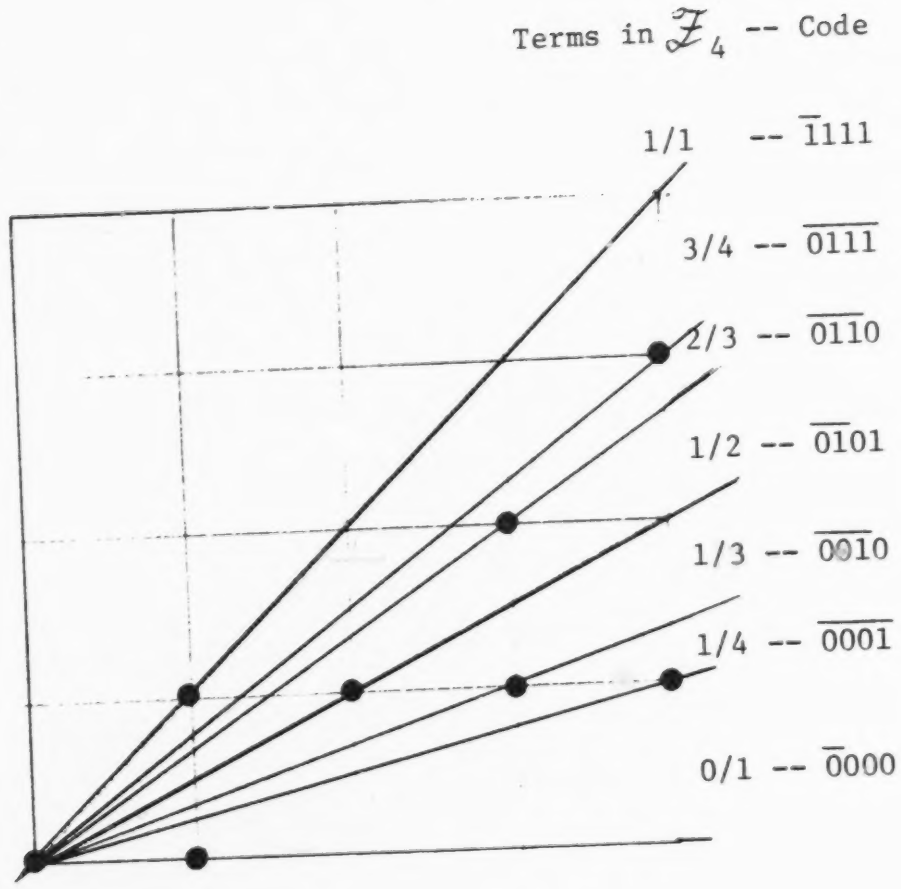


Figure II.4: Straight Line Code and Farey Series

is given in the following:

Theorem II.9: The number of classes of lines through the origin with slope between 0 and 1 on an nxn grid (or the number of distinct n-digit line codes, or the number of wedges with no interior lattice points) is roughly proportional to grid area, i.e., $\approx (3n^2)/\pi^2$.

Proof: Theorem II.8 gives the 1-to-1 correspondence between n-digit codes and $\mathcal{F}_n$. Theorem 331 on page 268 of Hardy and Wright states "The number of terms in the Farey series of order n is approximately $(3n^2)/\pi^2$."

Theorem II.10: The slope interval delimited by each wedge corresponding to a distinct n-digit code cannot be less than $1/n^2$ nor greater than $1/n$.

Proof: Theorem 28 on page 23 of Hardy and Wright states that if h/k and h'/k' are two successive terms in $\mathcal{F}_n$, then

$$kh'-hk'=1.$$

This may be rewritten as the difference between two successive terms

$$\frac{h'}{k'} - \frac{h}{k} = \frac{1}{kk'}$$

which is the slope interval delimited by a wedge. Since

$$k' \leq n \quad \text{and} \quad k \leq n, \quad \frac{1}{kk'} \geq \frac{1}{n^2} \quad .$$

Theorem 29 on page 23 states that if h/k, h''/k'', and h'/k' are three successive terms of $\mathcal{F}_n$, then

$$\frac{h''}{k''} = \frac{h+h'}{k+k'} \quad .$$

Thus $k+k' > n$ if h/k and h'/k' are two successive terms (otherwise

$\frac{h+h'}{k+k'}$, rather than h'/k', would be the successor of h/k). Thus a lower bound for kk' is

$k(n+1-k)$, where $0 < k \leq n$.

This quantity has minimum value n when k=1 or n. Thus

$$\frac{1}{kk'} \leq \frac{1}{n} .$$

In summary, finite grids yield only codes of lines of rational slopes, and these partition the slopes of lines fairly uniformly. The resolving power expressed by this partition is proportional to grid area. Since the codes of lines of rational slope are periodic, a single period is sufficient to describe the entire code of a line on a grid. The standard form of such a period (corresponding to the line through the origin) is found by cyclically permuting the digits of a period until a code which has minimum value when interpreted as a binary number results. Because of the above properties, all codes referred to in Chapter III will be single periods in standard form of code of lines with rational slopes.

CHAPTER III: AFFINE TRANSFORMATIONS AND STRAIGHT LINE CODES

Not all n-digit strings are the codes of straight lines on an n x n grid. There are constraints on line codes such as the homogeneous distribution of 1's described in theorem II.2. One way to recognize line codes is to apply geometric transformations to the grid that preserve straight lines but simplify their codes to a more recognizable form (e.g., an arbitrary number of 0's followed by a 1). If only lines are mapped into lines then only codes corresponding to cell configurations compatible with straight lines would be reduced to recognizable form. If code digits are expressed as cell states in a cellular automaton these code simplifications would be expressed as interactions between cells. Transformations that map lines into lines, preserving parallelism, on the plane are called affine in mathematics (see Rektorys (1969), p.228, for definition) and correspond to multiplying the coordinate vectors of points by a 2 x 2 matrix of constants with non-zero determinant and adding a constant vector. The latter vector will be considered (0,0) here because translation of a line may be ignored in code recognition since it introduces trivial code changes.

III.1 Unimodular Transformations

The unimodular (see Hardy & Wright (1965) p.28 for definition and properties) subgroup of the affine transformations maps lattice points into lattice points. Thus, unimodular transformations can be applied by simply relabelling lattice points without altering the grid. This

relabelling can be expressed in the cellular automata as simply re-defining what is to be considered as the set of nearest neighbors of each cell. Applying a unimodular transformation corresponds to multiplying the coordinate vectors of points by a 2 x 2 matrix of integers with determinant ± 1.

Two kinds of unimodular transformations that result in simple code changes are the shearing transformations S_k and the slope-complementing transformation C. Their respective matrices

$$S_k = \begin{pmatrix} 1 & 0 \\ k & 1 \end{pmatrix} \quad \text{and} \quad C = \begin{pmatrix} 1 & 1 \\ 0 & -1 \end{pmatrix}$$

are to postmultiply the row vectors of point coordinates.

Looking first at the shearing transformations, note

$$S_k S_j = S_{k+j} \quad \text{implying} \quad (S_k)^{-1} = S_{-k} .$$

Thus, the shearing transformations are isomorphic to the integers under addition. The original basis of the lattice is the pair of vectors from the origin to (1,0), and (0,1), i.e., the unit horizontal and vertical vectors respectively, starting at the origin. Applying S_k to this basis transforms it into the pair of vectors from the origin to (1,0) and (k,1) respectively. Thus the point whose coordinates in the old basis are (x,y) has coordinates:

$$(x,y)(S_k)^{-1} = (x,y)\begin{pmatrix} 1 & 0 \\ -k & 1 \end{pmatrix} = (x-ky,y)$$

with respect to the new basis. Note that the y-coordinate is unchanged,

but the x-coordinate is displaced ky units to the left, thereby shearing the grid. This basis change can be regarded as a change in the perspective from which the lattice is viewed. That is, the points are not moved but their labels are changed.

The code of a line of slope p/q through the origin has 1's in places corresponding to lattice columns in which the line crosses lattice horizontals. Substituting y=j into the equation of the line y = (p/q)x, the coordinates of the jth such crossing (the 0th being the origin), corresponding to the jth 1 in the code, are ((q/p)j,j). The coordinates of this point with respect to the new basis, resulting from the application of S_k to the old basis, are ((q/p)j-kj,j). That is, there are k fewer vertical columns of unit width between any two adjacent crossings of lattice horizontals by the line and hence k fewer 0's between adjacent code 1's, (k is always chosen to not exceed the number of 0's in the shortest code segments). Figure III.1 illustrates the effect of S_1 on the lattice and code for a line of slope 2/5. In summary:

Theorem III.1: Applying S_k to the lattice basis deletes k 0's from each segment of the code of a line on that lattice. Applying S_{-h} inserts h 0's in each segment, where h may be any positive integer but k must not exceed the number of 0's in any short segment of the code.

The deletion of 0's described above simplifies a code by decreasing the number of digits in it. Since the same number of 0's is deleted from each segment of a code by applying S_k there is no change in the difference between any two segment lengths. Thus the fact that a code satisfies or violates the segment length constraint of theorem II.2 is not altered by applying shearing transformations.

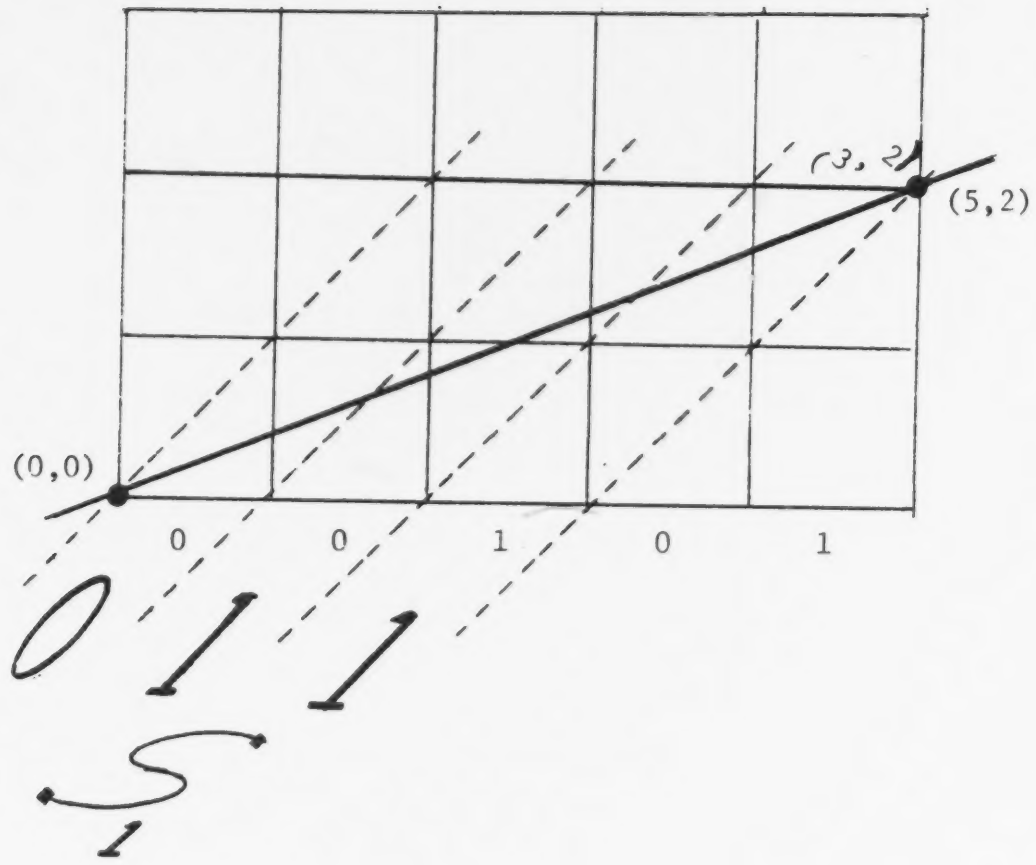


Figure III.1: Effect of Shearing Transformation on Code

The code change resulting from the application of S_k to a lattice can be effected in cellular automata by considering the cell one unit above and k units to the right of any cell as its upper neighbor. The bend configurations resulting from this redefining of local neighborhoods will then be separated by k fewer runs than under the usual definition of neighborhoods. Further discussion will be given when specific designs for cellular automata are presented in Chapter V.

Suppose the length of any segment in a code is either s or s-1. Applying S_{s-2} deletes the maximum number of 0's (i.e., s-2 per segment) possible from the code. Since its long and short segments then consist of 01 and 1 respectively, applying the slope-complementing transformation, whose effect on the code is as follows, is needed to return to the situation in which further 0-deletion is possible.

Theorem III.2: Applying C to the lattice basis interchanges 1's and 0's in the code of a line on the lattice except for the first and last digit of each period which remain 0 and 1 respectively.

Proof: Suppose the line goes through the origin and (q,p). Then applying C, which is its own inverse, to the lattice yields a line whose slope is the complement with respect to 1 of the original line, i.e.,

$$(q,p)\begin{pmatrix} 1 & 1 \\ 0 & -1 \end{pmatrix} = (q,q-p) \quad \text{or} \quad y = \frac{q-p}{q}x = (1-\frac{p}{q})x.$$

Thus the period of the code of the new line is also q and the slope is still between 0 and 1. Since the code digits in columns immediately following and preceding a lattice point crossed by

the corresponding line are always 0 and 1 respectively, the first and last digits of each period of the new code are 0 and 1 respectively.

Since the slope of the new line is (q-p)/q each period of its code contains as many 1's (i.e., q-p) as there are 0's in each period of the code of the old line. Equivalently, there are as many 0's in each period of the new code as there are 1's in each period of the old. Thus, the theorem can be proved if it can be shown that none of the 1's in the new code lie in the same columns as 1's of the old, i.e., 1's in one code correspond to 0's of the other except for the first and last digits of each period.

Assume the contradiction of the above, i.e., the jth 1 of the old code and kth 1 of the new both occupy the q_1th column of their respective periods. By the paragraph before Theorem III.1, this can be expressed as:

$$\lceil kq/(q-p) \rceil = \lceil jq/p \rceil = q_1$$

where $\lceil x \rceil$ means the least integer exceeding or equal to x.

Then, $jq = q_1 p - r_1 \quad ; 0 \leq r_1 < p$,

$kq = q_1(q-p) - r_2 ; 0 \leq r_2 < (q-p)$

Adding these equations and dividing by q yields

$$k+j = q_1 - (r_1+r_2)/q$$

Thus $(r_1+r_2)/q$ must be an integer, whence $r_1 = r_2 = 0$. This implies

that $q_1 = q$. But this value of q_1 corresponds to the 1 at the end of the code period which has already been considered.

The effect of C on the code can be expressed without regarding the first and last digits of a period as exceptions to the 0-1 interchange. Recall from Theorem II.6 that except for these two digits, a period of any code reads the same in forward or reverse order. Thus, reversing the order of a period of code makes the first digit 1 and the last 0 without changing the intervening digit. Hence, applying C yields the same result as reversing the order of each period and interchanging 0's and 1's.

III.2 Algorithms for Recognition of Straight Line Codes

In a line code whose long and short segments consist of 01 and 1 respectively, no 0 can be adjacent to another 0. Therefore, after applying C no 1 can be adjacent to another 1. That is, every code segment must contain at least one 0. Thus, a shearing transformation can be applied to shorten this code again into a code whose long and short segments consist of 01 and 1 respectively. Obviously this repetition of shearing and slope complementing transformations successively shortens the code. Eventually, the original line code will be reduced to a single digit, 1 if the last transformation is S_k and 0 if it is C. Summarizing the above and proving the converse is:

Theorem III.3: Applying the following algorithm to a period of any straight line code in standard form reduces it to a single digit (0). Binary strings not corresponding to straight lines are not reduced to a single digit.

1. Delete s-2 0's from each segment of the string, where s is the length of the longest segments in the string.

2. Reverse the order of the string and change all 0's to 1's and 1's to 0's.

Repeat steps 1 and 2 as long as the resulting string is more than one digit long and does not violate the segment length constraint of Theorem II.2. In case of such a violation, the string cannot be a period of straight line code and is rejected. If such a violation never occurs, the string is eventually reduced to the digit 0 and the original string constitutes a period of the code of a straight line.

Proof: It was shown in the paragraph preceding the theorem that a period of straight line code is reduced to a single digit by repeating steps 1 and 2.

To prove that any code that is reduced to a single digit by repeating steps 1 and 2 corresponds to a line, apply the inverse of those steps in reverse order starting with the string consisting of 0. The inverse of step 1 is the insertion of s-2 0's in each segment and corresponds to applying the affine transformation S_{2-s} to the lattice. Step 2 is its own inverse and corresponds to applying the affine transformation C to the lattice. The string consisting of 0 corresponds to a straight line through the origin with slope zero. Hence, applying the inverse of the steps that reduced the original code corresponds to applying affine transformations to the lattice, mapping the line of slope 0 into a line whose code is the original string.

Since the grid cannot resolve pattern loci within grid cells any number of patterns that are not straight lines can yield excited cell

configurations whose codes are those of straight lines. However, a code that is reduced to a single digit corresponds to a configuration of cells identical to that which would be excited by a straight line. Thus, the recognition algorithm of Theorem III.3 (and any algorithm based on the code) is really answering the question: "Can a straight line be drawn through all the cells excited by the pattern and no others?"

Any algorithm that reduces a period of straight line code to a single digit and rejects codes that cannot correspond to a line will be called a <u>reduction</u>. The algorithm of Theorem III.3 will be called the S-reduction because the shearing transformation S_k reduces the number of digits in the string. Since S_k and C are affine transformations, any sequence of inserting or deleting the same number of 0's in all segments or complementing digits and reversing string order, that results in a string consisting of a single digit, is a valid reduction. However, combining terms in any such reduction yields the sequence of Theorem III.3. That is, the affine transformations corresponding to any reduction can always be written as a sequence in which C and S_k alternate, i.e.,

$$(S_{k_1})CS_{k_2}CS_{k_3}\ .....S_k(C)$$

If there were any consecutive C's in the sequence they would cancel according to the rule

$$C^{(2n+1)}=C, \qquad C^{2n}=I=\begin{pmatrix}1 & 0\\ 0 & 1\end{pmatrix}.$$

Consecutive S_k's would combine according to the rule $S_k S_j = S_{k+j}$. Apparently, then, there is only freedom to choose the value of the subscripts and not the order of the transformations. Recall that the subscript cannot exceed s-2, the number of 0's in the shortest segments of the code. Therefore, suppose some subscript k less than s-2 is chosen. Applying S_k then yields a code with at least one 0 in each of its short segments and at least two 0's in each of its long segments. Applying C next yields a code with consecutive 1's occupying the places of the consecutive 0's in long segments of the code prior to the application of C. Thus, short segments of the code after C has been applied consist of 1's; there are no 0's that can be deleted so shearing transformations cannot be applied. Hence, C must be applied again; these two consecutive C's cancel, so k cannot be less than s-2.

Though the sequence of shearing and slope-complementing transformations that reduce the code of a particular line to a single digit is unique, terms may be combined in it to yield code reduction steps that are not expressed as the deletion and interchanging of digits. Since S_0, C^2 and CS_0C all equal the identity, they may be inserted anywhere in the sequence of transformations without changing the code. Also, S_{j+k} can be decomposed into $S_j S_k$ for purposes of recombining terms. The following theorem expresses a reduction algorithm based on such a combining of terms. The sequence of codes resulting from its application reveal some very important properties of the code, particularly the effects of translation and rotation.

Theorem III.4: Applying the following algorithm to a period of any straight line code in standard form reduces it to a single digit. Binary strings not corresponding to straight lines are not reduced to a single digit.

1. Replace each long segment in the code with a 0 and each short segment with a 1.

Repeat step 1 as long as the resulting string is more than one digit long and does not violate the segment length constraint of Theorem II.2. In case of such a violation the string is rejected. If such a violation never occurs the string is eventually reduced to the digit 0 and the original string constitutes a period of the code of a straight line.

Proof: Since whole segments are being rewritten as single digits the code is being shortened each time step 1 is applied. Step 1 corresponds to applying the sequence of affine transformations

$$S_{s-2}\, CS_1C \quad \text{(where s is the length of long segments)}$$

to the lattice. That is, S_{s-2} results in a code whose long and short segments are 01 and 1 respectively. Since S_1 removes one 0 from each segment CS_1C removes one 1 from each set of adjacent 1's. If this 1 is removed from the left end of each set of adjacent 1's it is the 1 in the segment 01. Thus, after the application of S_{s-2}, CS_1C replaces long segments (01) by 0 and short segments (1) by 1.

Since this algorithm corresponds to the application of a sequence of affine transformations, a code that is reduced to a single digit by the algorithm must be a period of the code of a straight line by the same argument as in the proof of Theorem III.3.

In each rewriting step of Theorem III.4, long segments are treated as horizontals and short segments as lines of slope 1 in the new perspective corresponding to the affine transformation of the lattice. It will be shown in Chapter IV that the slope of the line in this new perspective is the complement with respect to 1 of the remainder of the reciprocal of the original slope quotient. Successive rewriting steps then lead to a continued fraction development of the original slope quotient. For this reason, the algorithm of Theorem III.4 is called the CCF-reduction (Complemented Continued Fraction). In an analogous manner the rewriting steps of the following theorem lead to the ordinary continued fraction development of the slope quotient, different from that of Theorem III.4. For this reason, the algorithm of Theorem III.5 is called the CF-reduction (Continued Fraction).

Theorem III.5: Applying the following algorithm to a period of any straight line code in standard form reduces it to a single digit. Binary strings not corresponding to straight lines are not reduced to a single digit.

1. Replace each long segment in the code with a 1 and each short segment with a 0, then reverse the order of the string.

Follow the same instructions as appear after step 1 in Theorem III.4.

Proof: The code resulting from applying step 1 in this theorem differs from the code resulting from step 1 in the previous theorem in that the roles of 1's and 0's are interchanged and the order is reversed. That is, a C transformation has been added here so the affine transformation corresponding to

step 1 is:

$$S_{s-2}\ CS_1C\cdot C = S_{s-2}\ C\ S_1$$

Thus, the code is shortened at each step by the application of an affine transformation and the theorem may be proved by the same argument as in Theorem III.4.

In the following chapter code properties revealed by the CCF-reduction and the related CF-reduction will be discussed in detail and the interesting interplay between Farey series, continued fractions and straight line codes will be presented.

CHAPTER IV: CODE TRANSFORMATIONS, CONTINUED FRACTIONS AND FAREY SERIES

IV.1 Code Reduction and Continued Fractions

Since the code of a line whose slope is p/q has p 1's per period, and thus p segments, rewriting long and short segments as 0's and 1's respectively yields a code with p digits per period. This latter code, resulting from the application of one step of the CCF-reduction (Theorem III.4), will be called the new code and the code for the line of slope p/q the old code in the following discussion. The number of 1's in the new code is the number of short segments in the old. But this is just the number of 0's that would have to be inserted in the old code to yield a code composed of only long segments. Call this number of 1's in the new code p_2'. The slope of the line corresponding to such a code consisting of only long segments is $p/(q+p_2')$. Its reciprocal, $(q+p_2')/p$, is an integer equal to the length of these long segments. From the proof of Theorem III.2, the leftmost 1 in the old code occurs in column $\lceil q/p \rceil$. This 1 terminates the first segment, which must be long since rewriting it yields 0 as the first digit of the new code. The length of long segments in the old code is thus $\lceil q/p \rceil$, i.e.,

$$(q+p_2')/p = \lceil q/p \rceil = \lfloor q/p \rfloor + 1$$

Here $\lfloor q/p \rfloor$ is the greatest integer not exceeding q/p. Equivalently

$$q/p - \lfloor q/p \rfloor = (p-p_2')/p = r/p$$

where r is the remainder on division of q by p, or

$$q = \lfloor q/p \rfloor \cdot p+r = \lceil q/p \rceil \cdot p-p_2' \ , \quad 0 \leq r < p, \ 0 \leq p_2' < p$$

and

$$p_2' = p-r.$$

The new code thus corresponds to a line of slope p_2'/p, the complement with respect to 1 of the remainder of the quotient of q by p. The slope of the old line can now be written

$$p/q = \frac{1}{\lceil q/p \rceil - p_2'/p} \ .$$

Rewriting the long and short segments of the new code as 0 and 1 respectively leads to a similar expression for p_2'/p which can be incorporated into the expression for p/q as

$$p/q = \cfrac{1}{\lceil q/p \rceil - \cfrac{1}{\lceil p/p_2' \rceil - p_3'/p_2'}} \ .$$

Carrying out the entire CCF-reduction on the old code then leads to the continued fraction

$$p/q = \cfrac{1}{\lceil q_1'/p_1' \rceil - \cfrac{1}{\lceil q_2'/p_2' \rceil - \cfrac{1}{\lceil q_3'/p_3' \rceil - \ddots - \cfrac{1}{\lceil q_N'/p_N' \rceil}}}}$$

where p'_j/q'_j is the slope of the line corresponding to the code prior to the jth rewriting of segments as digits. From the preceding, it is apparent that the coefficient $\lceil q'_j/p'_j \rceil$ of the CCF is the length of long segments of code prior to the jth rewriting of segments as digits. Calling this segment length s_j, the continued fraction will be written

$$p/q = \lceil s_1, s_2, s_3, \ldots . s_N \rceil .$$

Consider now the result of applying one step of the CF-reduction (Theorem III.5) to the code of a line of slope p/q. As in the CCF-reduction, the number of digits in the new code is p, the number of 1's in the old code. However, since the role of 1's and 0's is interchanged when applying the CF-reduction rather than the CCF-reduction, the number of 1's in the new code is $p-p'_2 = r$, the remainder of the quotient of q by p. By the same process as just described for the CCF-reduction, the ordinary continued fraction, with positive rather than the negative signs resulting from the CCF reduction, can be developed using the Euclidean division algorithm (see Hardy and Wright, 1965, p. 136), i.e.,

$$p/q = \cfrac{1}{\lfloor q/p \rfloor + \cfrac{1}{\lfloor w'_2/v'_2 \rfloor + \cfrac{1}{\lfloor w'_3/v'_3 \rfloor + \cfrac{1}{\ddots + \cfrac{1}{\lfloor w_M/v_M \rfloor}}}}}$$

Here v'_j/w'_j is the slope of the line corresponding to the code prior to

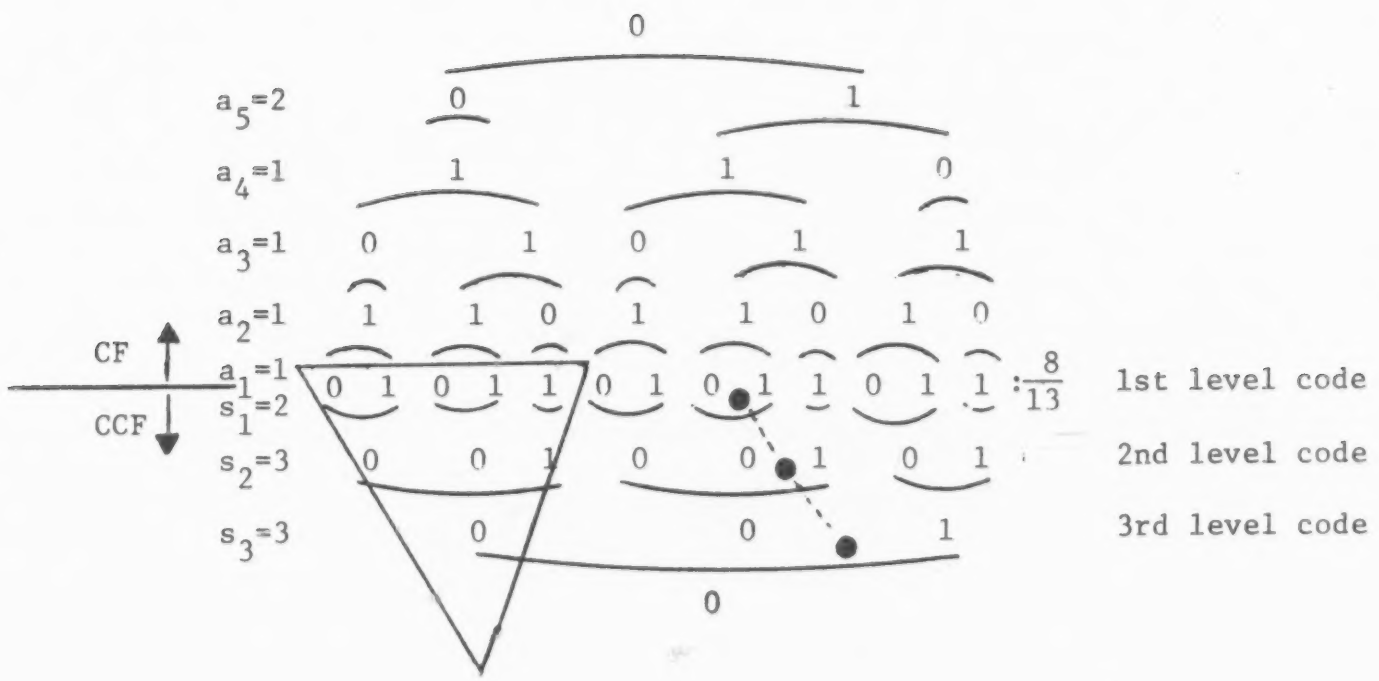
the jth rewriting of long segments as 1's and short segments as 0's. The length of short segments of code prior to the jth rewriting is $\lfloor w'_j / v'_j \rfloor$. Calling this length a_j, the ordinary continued fraction (CF) will be written

$$p/q = \lfloor a_1, a_2, a_3, \ldots a_M \rfloor .$$

Figure IV.1 illustrates the CF- and CCF-reduction of the code of a line of slope 8/13 and the corresponding continued fractions. Obviously, both reduction processes can be reversed to generate the code of a line of any rational slope p/q between 0 and 1. That is, derive the CF or CCF by applying to p/q the Euclidean division algorithm or the division with negative remainder, respectively. In the case of the CF, write a segment of length a_M and then rewrite its 0's as segments of length a_{M-1}-1 and its 1's as segments of length a_{M-1} and reverse the order of the string. Repeat this rewriting of 0's as short segments and 1's as long segments and reversing digit order using a_i's with successively decreasing subscripts to determine the length of short segments for successive rewritings. After all a_i's have been used, the result is the code of a line of slope p/q. The same code can be generated using the CCF for p/q by starting with segment of length s_N and successively rewriting 0's as long segments and 1's as short segments using s_j's with decreasing subscripts to determine the length of long segments for each rewriting.

Although the CF and CCF share many important properties, they have some interesting differences. The CCF yields much clearer

$$\frac{8}{13} = \cfrac{1}{1 + \cfrac{1}{1 + \cfrac{1}{1 + \cfrac{1}{1 + \frac{1}{2}}}}}$$



$$\frac{8}{13} = \cfrac{1}{2 - \cfrac{1}{3 - \frac{1}{3}}} \quad ; \quad \frac{p_1}{q_1} = \frac{1}{2} \quad , \quad \frac{p_2}{q_2} = \frac{3}{5} \quad , \quad \frac{p_3}{q_3} = \frac{8}{13}$$

Figure IV.1: Code Reduction and Continued Fractions

derivations of code properties and their interpretation in terms of grid geometry than the CF. The following discussion and theorems express some of the properties of CF and CCF and their relevance to the code and grid geometry.

The large body of mathematical literature on continued fractions contains full treatments of their use in approximating the values of complicated functions by rational numbers. Thorough presentations of such topics and good bibliographies can be found in Perron (1960) Wall (1948), Khovanskii (1963), and Sierpinski (1964). Here, elementary properties of CF's and CCF's and their relation of grid geometry are of interest; many of these for CF's are in Hardy and Wright (1965). Relevant theorems from there will be given here without proofs and labelled by their numbers in Hardy and Wright (1965) followed by HW. Some of their notation will be changed to avoid conflict with notation used here. For example, their CF's have a 0th term, i.e.,

$$p/q = \lfloor a_0, a_1, \ldots . a_M \rfloor$$

so that all positive rational numbers can be represented. Here only slopes between 0 and 1 are considered so $a_0 = 0$ always, and is therefore omitted. Since CCF's are generally neglected in the literature except for brief statements of elementary theorems, proofs here will be given in detail for theorems concerning them.

The fractions

$$v_i/w_i = \lfloor a_1, a_2, \ldots . a_i \rfloor \qquad 1 \leq i \leq M$$

and

$$p_j/q_j = \lceil s_1, s_2, \ldots s_j \rceil \qquad 1 \leq j \leq N$$

are called the ith and jth convergents to the CF and CCF respectively. In this context, the notation $\lfloor a_1 \rfloor$ or $\lceil s_1 \rceil$ means the fraction $1/a_1$ or $1/s_1$ which conflicts with the notation $\lfloor\ \rfloor$ and $\lceil\ \rceil$ used in Chapter III. This ambiguity will be resolved either explicitly or by context when it appears. In any case, the brackets will seldom contain a single term when used to denote a convergent to a continued fraction. The following two theorems express recursion rules for deriving successive convergents.

Theorem 149HW: If v_i and w_i are defined by

$$v_1 = 1, \quad v_2 = a_2, \ldots, \quad v_i = a_i v_{i-1} + v_{i-2}$$

$$w_1 = a_1, \quad w_2 = a_1 a_2 + 1, \ldots, w_i = a_i w_{i-1} + w_{i-2}$$

then $\lfloor a_1, a_2, \ldots a_i \rfloor = v_i/w_i$.

Similarly,

Theorem IV.1: If p_j and q_j are defined by

$$p_1 = 1, \quad p_2 = s_2, \ldots, \quad p_j = s_j p_{j-1} - p_{j-2}$$

$$q_1 = s_1, \quad q_2 = s_1 s_2 - 1, \ldots, \quad q_j = s_j q_{j-1} - q_{j-2}$$

then $\lceil s_1, s_2, \ldots s_j \rceil = p_j/q_j$

Proof: (By induction) For j=3, $\lceil s_1, s_2, s_3 \rceil = \cfrac{1}{s_1 - \cfrac{1}{s_2 - \cfrac{1}{s_3}}}$

$$= \frac{1}{s_1 - \dfrac{s_3}{s_2 s_3 - 1}} \qquad = \frac{s_2 s_3 - 1}{s_1 s_2 s_3 - s_1 - s_3}$$

$$= p_3/q_3 \quad .$$

Now suppose the theorem holds for the j<u>th</u> case and write the (j+1)<u>th</u> convergnet as $\lceil s_1, s_2, \ldots, s_j, s_{j+1} \rceil = \lceil s_1, s_2, \ldots, s_j - \frac{1}{s_{j+1}} \rceil$, i.e., as a CCF with j levels. Then,

$$\lceil s_1, \ldots, s_j - \frac{1}{s_{j+1}} \rceil = \frac{(s_j - 1/s_{j+1})p_{j-1} - p_{j-2}}{(s_j - 1/s_{j+1})q_{j-1} - q_{j-2}} =$$

$$= \frac{s_{j+1}s_j p_{j-1} - s_{j+1}p_{j-2} - p_{j-1}}{s_{j+1}s_j q_{j-1} - s_{j+1}q_{j-2} - q_{j-1}} = \frac{s_{j+1}p_j - p_{j-1}}{s_{j+1}q_j - q_{j-1}}$$

which completes the proof.

The preceding can be used to prove the following theorems.

<u>Theorem 150HW</u>: $v_i w_{i-1} - v_{i-1} w_i = (-1)^{i-1}$ or

$$(v_i/w_i) - (v_{i-1}/w_{i-1}) = (-1)^{i-1}/(w_i w_{i-1})$$

<u>Theorem 151HW</u>: $v_i w_{i-2} - v_{i-2} w_i = (-1)^i a_i$ or

$$(v_i/w_i) - (v_{i-2}/w_{i-2}) = (-1)^i a_i/(w_{i-2}w_i)$$

It follows from 149HW that all v_i, w_i are positive, so 150 HW implies that the sign of the difference between two successive convergents alternates. Theorem 151HW implies that the sequence of even convergents increases monotonically and the sequence of odd convergents decreases monotonically. Since $p/q = v_M/w_M$ (the final convergent) its value must exceed that of any even convergent and be less than any odd convergent. Thus, the value of each successive odd/even convergent is closer to p/q than all preceding odd/even convergents. (See Hardy and Wright, 1965, p. 132 for details).

If the CF convergents are regarded as slopes, the geometrical interpretation of 150HW and 151HW is that the slopes alternately exceed and are exceeded by p/q but the difference in slope between p/q and successive slopes (convergents) decreases. Theorems 155HW and 157HW state that $w_i > w_{i-1}$ and v_i/w_i is in lowest terms. Therefore, there are no lattice points between the origin and (w_i, v_i) on the line connecting them and the distance between the origin and successive points corresponding to CF convergents increases monotonically. Figure IV.2 illustrates these properties.

Convergents to the CCF behave differently, i.e.,

Theorem IV.2: $p_j q_{j-1} - q_j p_{j-1} = 1$ or $(p_j/q_j) - (p_{j-1}/q_{j-1}) = 1/(q_j q_{j-1})$

Proof: $p_j q_{j-1} - q_j p_{j-1} = s_j p_{j-1} q_{j-1} - p_{j-2} q_{j-1} - s_j q_{j-1} p_{j-1} + p_{j-1} q_{j-2} =$

$= p_{j-1} q_{j-2} - p_{j-2} q_{j-1} = \cdots\cdots = p_2 q_1 - q_2 p_1 = s_1 s_2 - (s_1 s_2 - 1) = 1.$

The difference between the jth and (j-1)th convergents is always positive so the sequence of CCF convergents increases monotonically. Since $p/q = p_N/q_N$ (the final convergent) its value exceeds that of all other convergents and each convergent is closer to p/q than all those preceding. Analogous to 151HW is:

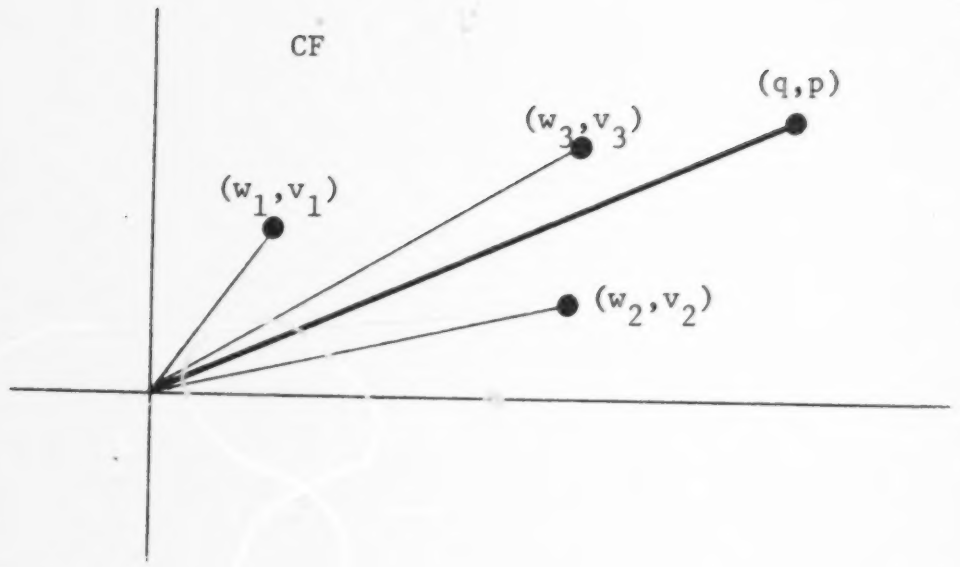
Theorem IV.3: $p_j q_{j-2} - p_{j-2} q_j = s_j$ or $(p_j/q_j) - (p_{j-2}/q_{j-2}) = s_j (q_{j-2} q_j)$

Proof: $p_j q_{j-2} - p_{j-2} q_j = s_j p_{j-1} q_{j-2} - p_{j-2} q_{j-2} - s_j q_{j-1} p_{j-2} + p_{j-2} q_{j-2} =$

$= s_j (p_{j-1} q_{j-2} - q_{j-1} p_{j-2}) = s_j$.

The following three theorems lead to a geometric interpretation of CCF convergents analogous to that for CF convergents.

Theorem IV.4: $s_j \geq 2$.



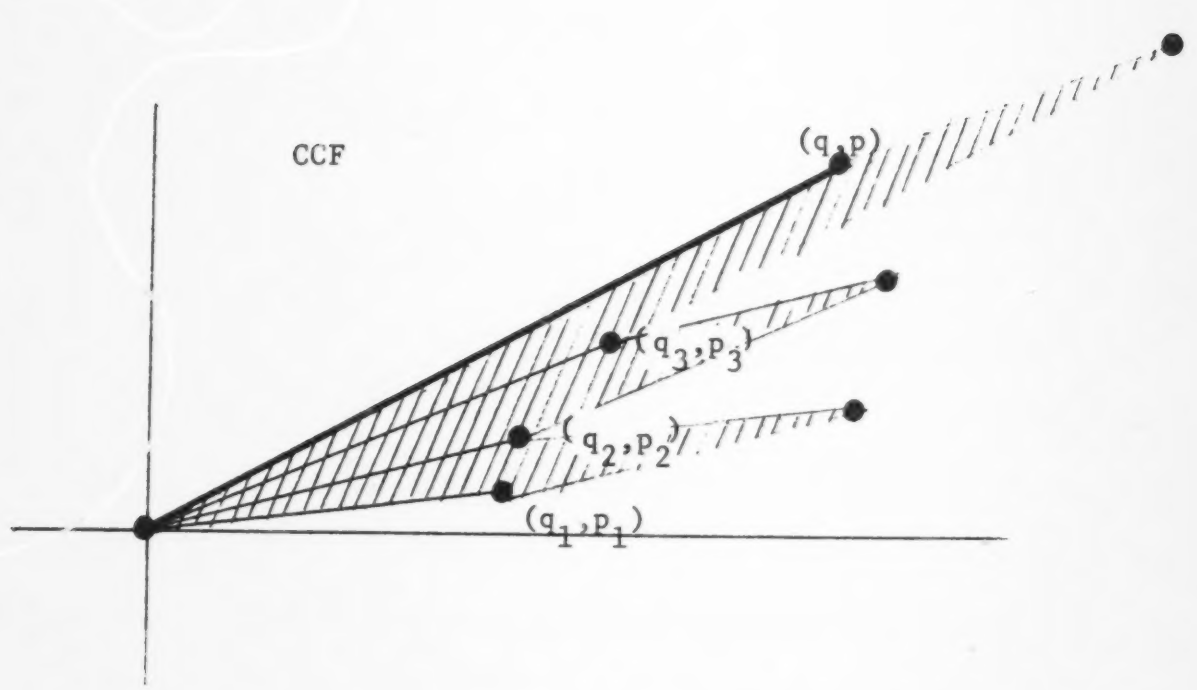


Figure IV.2: Geometric Interpretation of CF and CCF convergents

Proof: The (j-1)th CCF reduction step corresponds to generating the code of the complemented remainder p'_j/q'_j of the quotient q'_{j-1}/p'_{j-1} as in the discussion at the beginning of IV.1. Since this remainder is less than 1, its reciprocal is greater than 1, so

$$s_j = \lceil q'_j/p'_j \rceil \geq q'_j/p'_j > 1, \qquad \text{Hence } s_j \geq 2.$$

Theorem IV.5: $q_j > q_{j-1}$, $p_j > p_{j-1}$.

Proof: (By induction)

$$q_2 = s_1 s_2 - 1 \geq 2s_1 - 1 = s_1 + (s_1 - 1) \geq s_1 + 1 > s_1 = q_1$$

Now suppose $q_j > q_{j-1}$ then

$$q_{j+1} = s_{j+1} q_j - q_{j-1} \geq 2q_j - q_{j-1} = q_j + (q_j - q_{j-1}) > q_j$$

The same argument applies to p_j.

Theorem IV.6: p_j/q_j is in lowest terms.

Proof: Suppose there is a common divisor d, i.e., $d \mid p_j$ and $d \mid q_j$. Then $d \mid (p_j q_{j-1} - q_j p_{j-1})$. But from Theorem IV.2, the quantity in parentheses is 1, so $d \mid 1$ whence d=1.

Referring now to Figure IV.2, Theorem IV.6 can be geometrically interpreted as stating that there are no lattice points between the origin and (q_j, p_j) on the line connecting them. Theorem IV.5 implies that points corresponding to successive convergents are successively further from the origin, and Theorem IV.2 implies that the slopes of lines connecting the origin to those points are successively closer to p/q.

Convergents to the CF of any number, rational or irrational, are the best possible rational approximations to it in the following sense. If v_i/w_i is a CF convergent to m then there are no fractions v/w with

smaller denominator ($w \leq w_i$) such that $|v - wm| < |v_i - w_i m|$ (182 HW). The geometric interpretation of this CF property is that there are no points to the left of (w_i, v_i) that are closer than it is to the line $y = mx$. Convergents to the CCF of any number are the best rational approximation less than that number in the same sense as CF convergents. That is,

Theorem IV.7: If p_j/q_j is a CCF convergent to p/q there are no other fractions v/w, where $w \leq q_j$, $v/w < p/q$ satisfying

$$|v - w(p/q)| \leq |p_j - q_j(p/q)|$$

That is, the only lattice points beneath the line of slope p/q closer to it than (q_j, p_j) lie to the right of that point.

Proof: A geometric interpretation of Theorem IV.2 is that there are no lattice points in the interior or on the sides of the parallelogram (0, 0), (q_j, p_j), $(q_{j-1}+q_j, p_{j-1}+p_j)$, (q_{j-1}, p_{j-1}) formed by the lines connecting the origin to two successive convergents except for the vertices. These parallelograms free of lattice points are shaded in Figure IV.2. Furthermore, from Theorem IV.5 the leftmost new parallelogram vertices introduced by taking successive CCF convergents lie to the right of all previous such parallelogram vertices. From Theorem IV.2 the sides that are shared by two parallelograms are of successively increasing slope for successive convergents. The final parallelogram is bounded on one side by the line of slope p/q through the origin.

Thus, the entire region enclosed by the lines

$$y = (p/q)x, \quad y = (p_j/q_j)x, \quad x = q_j$$

is covered by parallelograms which, except for their vertices, are free of lattice points. The vertices of these parallelograms all lie to the right of (q_j, p_j) except for the vertex (0, 0) they all share. Figure IV.2 illustrates this situation.

An alternate proof of Theorem IV.7 may be derived from the code properties illustrated in Figure IV.1. Call the code resulting from the (j-1)th application of a CCF reduction step to the code for p/q the jth level code. That is, the 1st level code is the code for p/q and the Nth level code consists of a single segment of length s_N. Similarly, the jth level code for p_j/q_j consists of a single segment of length s_j. This segment corresponds to a single digit, 0, in the (j+1)th level code for p_{j+1}/q_{j+1}. Therefore, the 1st level code for p_{j+1}/q_{j+1} consists of the concatenation of $(s_{j+1}-1)$ periods of the 1st level code for p_j/q_j (each of which corresponds to a 0 in the (j+1)th level code) followed by some other code. This other code will be derived in IV.2; for now it suffices to know that the code for the (j+1)th CCF convergent is derived from the code for the jth convergent by concatenating at least one period of the latter with some other code. That is, none of the first q_j digits are altered. Therefore, there can be no lattice points to the left of (q_j, p_j) between the lines of slope p_j/q_j and p/q. If there were, a code 1 and 0 would interchange somewhere in the first q_j digits as described in the first paragraph of II.5 when the slope of the line is increased.

In Figure IV.1, the triangle encloses the 1st, 2nd, and 3rd level codes for p_2/q_2. The relation between concatenation and successive CCF convergents is thereby illustrated for p_2/q_2 and p_3/q_3. Although CF convergents may be derived from the codes resulting from CF reduction of the code for p/q, the reversal of code order at each step results in a more complicated code change than the simple concatenation shown for CCF convergents. That is, the code for v_i/w_i corresponds to a 1 at the (i+1)th level, and since this 1 is the first or last digit at the (i+1)th level depending on whether i is even or odd respectively, the derivation of codes for successive convergents involves generating new code from right to left as well as left to right. From Theorem IV.7, nothing is lost in considering CCF convergents rather than CF convergents, so the former will be used exclusively in the following text. The advantages of the CCF over the CF are the monotonicity of the sequence of convergents and the simple concatenation structure of the corresponding codes. In the rest of this chapter the CCF structure of the code will be used to derive code transformations related to translation and rotation of lines on the grid.

IV.2 Translation of Lines and Cyclic Shifts in Code

IV.2.1 Cyclic Shifts and Code Catenation

Recall from Theorem II.4 that upward translation of a line of slope p/q results in the right cyclic shift by q' digits of each period of the code for every 1/q units vertical displacement. This cyclic shift is effected by the transposition of a 0 and 1 when the line is

displaced enough to pass through lattice points closest to the line of slope p/q through the origin. Call the original and displaced lines L and L' as illustrated in Figure IV.3. Since there are no lattice points in the strip between L and L', the line L' and the line $y = (p'/q')x$ cross the same cell sides in the first q' columns, so the first q' digits of the code after such a cyclic shift constitute the code for the line $y = (p'/q')x$.

The code for p'/q' can be derived from the code for p/q by using the multilevel CCF structure of the code illustrated in Figure IV.1. Transposing a 0 and 1 in the q'th and (q'+1)th columns of the 1st level code yields the same result as transposing the short segment following the 1 and the long segment terminating in that 1 since the long segment is shortened by one digit and short segment is lengthened by one digit. This segment transposition in the 1st level code corresponds to the transposition of a 0 and a 1 in the 2nd level code exactly like the transposition in the 1st level code. In the same manner, the transposition propagates through all higher level codes. At the Nth level, such a transposition can only occur at the end of the code, i.e, the only 1, which terminates the Nth level code, is transposed with the 0 to its left. The points connected by the dotted line in Figure IV.1 lie between the digits to be transposed at each level.

The first q' digits of the 1st level code after the single transposition described in the preceding paragraph correspond to the first s_N-1 digits of the Nth level code. The lengths of long and short segments at all levels are unchanged by the transposition except in the

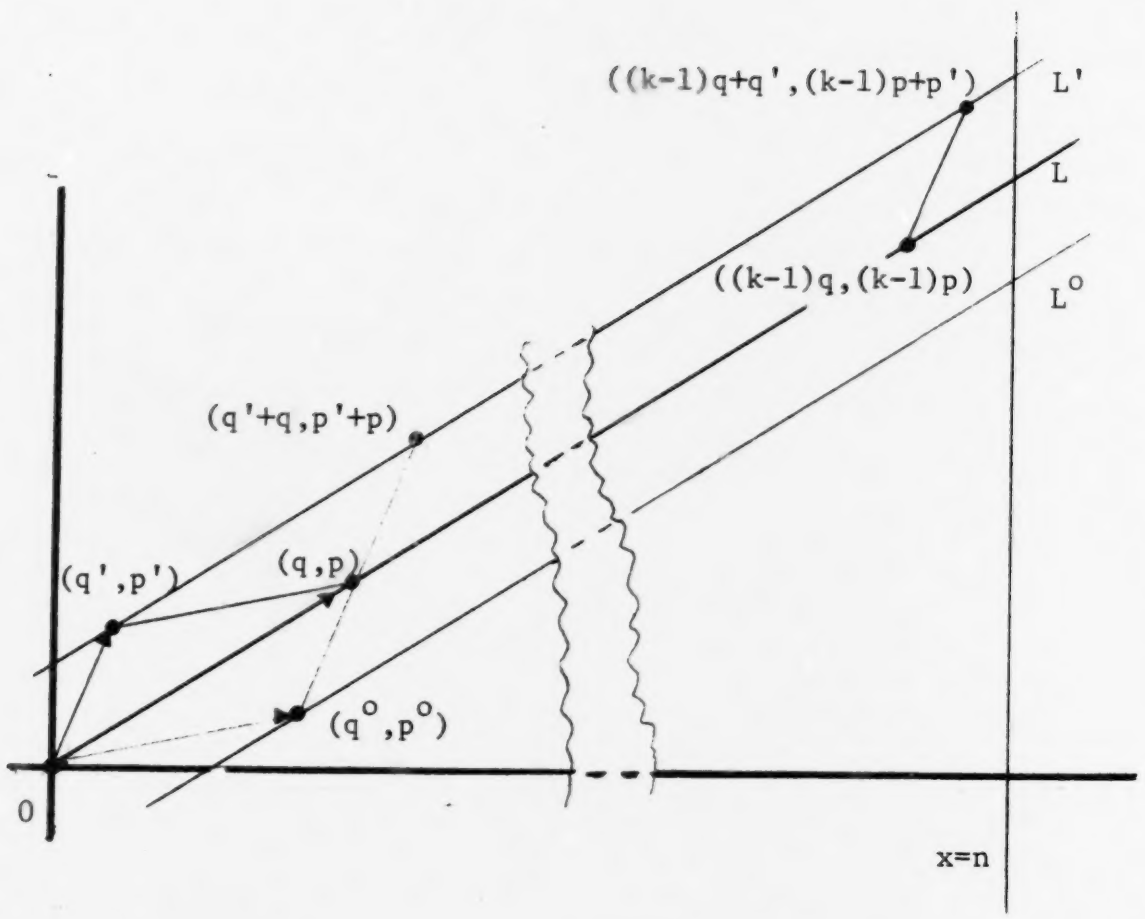


Figure IV.3: Lattice Geometry Related to Cyclic Shift in Code

Nth level where segment length has been decreased by one digit. Therefore, if the CCF for p/q is

$$\lceil s_1, \ldots . s_N \rceil$$

the CCF of p'/q' must be

$$\lceil s_1, \ldots, s_N - \overline{1} \rceil = \lceil s_1, \ldots, s_N, \overline{1} \rceil$$

Applying Theorem IV.1 to the latter CCF yields

Theorem IV.8: $q' = s_{N+1}q_N - q_{N-1} = q - q_{N-1}$, $p' = p - p_{N-1}$

That is, the magnitude of the cyclic shift is the denominator of the slope minus the denominator of the next to the last convergent to the CCF of the slope.

Since downward translation of a line of slope p/q results in left cyclic shifts by q' digits, the x coordinates of the points on the line L^o of closest lattice points below L in Figure IV.3 must be q-q'+kq, k = 0, $\pm$ 1, $\pm$ 2, ... Therefore the point (q^o, p^o) on L^o must be (q-q', p-q') which by Theorem IV.8 is (q_{N-1}, p_{N-1}). The geometry of the lattice line intersections with the line segment connecting (q', p') and (q,p) is congruent to the geometry of the lattice line intersections with the line segment connecting the origin to (q^o, p^o). Hence, the codes of these two line segments are identical. But the code for the former segment is just the (q'+1)th through qth digits of the code for L'. Therefore the first q digits of the code for L' are the concatenation of the code for p'/q' and the code for p^o/q^o. The first q digits of the code for L are the concatenation of the code for p^o/q^o and the code for p'/q' because the line segment connecting the origin and (q, p) crosses the same sides of lattice cells as the line segment from

the origin to (q^o, p^o) followed by the line segment from (q^o, p^o) to (q, p). That is, there are no lattice points within the triangle $(0, 0)$, (q^o, p^o), (q, p). In summary,

Theorem IV.9: The code for p/q is the concatenation of the codes for p_{N-1}/q_{N-1} and $(p-p_{N-1})/(q-q_{N-1})$. Upward translation of the line $y = (p/q)x$ by $1/q$ units reverses the order of the concatenation.

Notice that if the code for p/q had one more level (the (N+1)th level) the segment of length s_N-1 would be a short segment at the Nth level and the segment of length s_N would then be a long segment. These long and short segments at the Nth level would then correspond to 0's and 1's respectively at the (N+1)th level. Therefore, the 1st level code corresponding to a 1 at the (N+1)th level is simply the code for p'/q'. The discussion of the alternate proof of Theorem II.7 based on the code can now be completed with:

Theorem IV.10: The code for the CCF convergent p_{j+1}/q_{j+1} is derived from the code for p_j/q_j by concatenating s_{j+1}-1 periods of the latter with one period of the code for $(p_j-p_{j-1})/(q_j-q_{j-1})$.

Proof: The (j+1)th level code for p_{j+1}/q_{j+1} consists of (s_j-1) 0's followed by a 1. These 0's correspond to the 1st level code for p_j/q_j and the 1 corresponds to the 1st level code for p'/q' in Theorem IV.8 substituting j for N.

Note that Theorem IV.8 can be regarded as a special case of Theorem IV.10 where $j = N+1$ and $s_{N+1} = 1$. It is given as a separate theorem because it plays such an important role in the code transforma-

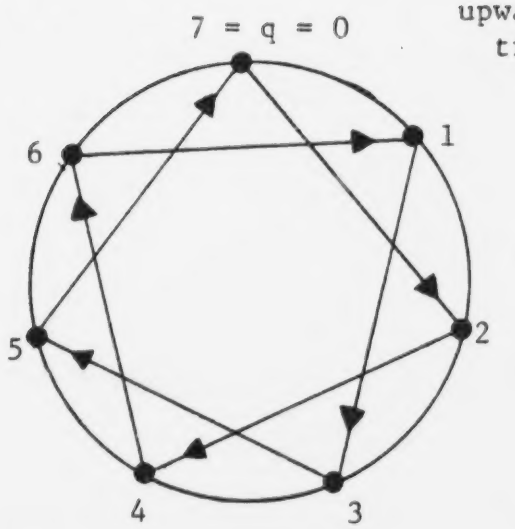
tions resulting from translation and rotation of lines.

IV.2.2 Shift Polygons

The cyclic shift of the code for p/q by q' digits to the right resulting from trnaslation of the corresponding line (q' will be referr to as a right shift for brevity) can be illustrated by constructing a "shift polygon" as follows. (Refer to Figure IV.4) Divide the circumference of a circle into q equal arcs with points labelled in the clockwise direction 0, 1, ..., q-2, q-1. The jth point represents a cyclic shift of j columns to the right in each period of the code for p/q. A succession of shifts is represented by a succession of directed chords linking the corresponding points. Thus, the shifts correspondin to upward translation by one unit distance of a line of slope p/q are represented by a q sided regular polygon (which may be re-entrant or star shaped) obtained by linking in order the points 0, q', 2q'mod(q), ...jq'mod(q), ..., qq'mod(q) = 0. For any q, p uniquely determines q' by Theorem IV.8. Moving the line downward instead of upward results in shifts of -q' digits each, corresponding to traversing the shift polygon in the opposite direction. In Figure IV.3 the lines L' and L^{o}, whose respective equations are

$$y = (p/q)x+(1/q) \quad \text{and} \quad y = (p/q)x-(1/q)$$

pass through the lattice points that determine the cyclic shifts in the code of the line L of slope p/q through the origin. The following theorem expresses a property of these points on L^{o} and L' which will be used to prove several theorems related to shift polygons.



shift polygon for 3/7

upward translation of line

vertex	shift
0	0 0•1 0 1 0 1
2	0 1 0 0•1 0 1
4	0 1 0 1 0 0•1
6	0•1 0 1 0 1 0
1	1 0 0•1 0 1 0
3	1 0 1 0 0•1 0
5	1 0 1 0 1 0 0•
7	0 0 1 0 1 0 1

$$\frac{p}{q} = \frac{1}{3 - \frac{1}{2 - \frac{1}{2}}} = \frac{3}{7} \quad , \qquad \frac{p'}{q'} = \frac{1}{3 - \frac{1}{2 - \frac{1}{2 - \frac{1}{1}}}} = \frac{1}{2}$$

$$\frac{q'}{q} = \frac{1}{4 - \frac{1}{2}} = \frac{2}{7} \quad , \qquad \frac{p'}{p} = \frac{1}{4 - \frac{1}{2 - \frac{1}{1}}} = \frac{1}{3}$$

Figure IV.4: Relation of Shift Polygon to Code and Continued Fraction

Theorem IV.11: All lattice points (w, v) on L' and L^o, satisfy the equation

$$vp-wq = \pm 1.$$

The sign is positive for points on L' and negative for points on L^o. No other lattice points satisfy this equation for a given q and p.

Proof: For a point (w, v) to satisfy the equation, the points (0, 0), (q, p), (w, v), (q+w, p+v) must be the vertices of a parallelogram of unit area. Since one base of this parallelogram is the line segment connecting (0, 0) and (q, p) whose length is $(p^2+q^2)^{1/2}$, the altitude must be $1/(p^2+q^2)^{1/2}$. But this altitude is the perpendicular distance of L from both L' and L^o. Therefore, (w, v) must lie on L' or L^o and since all points on these lines are the same perpendicular distance from L they all qualify as vertices of the unit area parallelograms in question. The cross product* in the equation is positive if the vector from the origin to (w, v) lies in a clockwise direction relative to the vector from the origin to (q, p) and negative if the positions are reversed.

Since (q', p') is on the line L' in Figure IV.3, its cross product with (q, p) is

$$p'q-q'p = 1.$$

But this is also the expression for the cross product of the points (p, p') and (q, q'). Since $p' < p < q$ and $p' < q' < q$, the point (p, p') must be the point determining the digit shift in the code of the line $y = (q'/q)x$. The right shift q' is a function of p and q, which can be written as

---

* Hereafter, the cross product of (x_1, y_1) and (x_2, y_2) is defined as the number $x_2y_1 - y_2x_1$.

$$P_q(p) = q'.$$

Paraphrasing the previous sentence in terms of it yields

$$P_q(q') = p \quad ,$$

whence

$$P_q(P_q(p)) = p \quad ,$$

or the function $P_q(p)$ is equal to its inverse. This proves the following theorem.

Theorem IV.12: If q' is the right shift in the code of a line of slope p/q, then p is the right shift in the code of a line of slope q'/q.

The preceding shows how the slope of a line can be determined from the shift polygon. The numerator of the slope is derived from q' (the number of vertex separated arcs subtended by each polygon side) by substituting the latter into the same function P_q that yields q' as a function of p. The denominator q of the slope is the number of polygon vertices by definition. The set of all integers between 0 and q that are prime to q is both the domain and range of P_q which is therefore a permutation on that set. Even powers of that permutation equal the identity.

In the left column of Figure IV.5 are all the fractions $0 < p/q < 1$ for $q = 7$. Opposite each of these fractions in the right column is the fraction corresponding to the lattice point that determines code shifts when the line of slope p/q is translated. The denominators on the right are the magnitude of the cyclic shift or the number of sides crossed by each side of the shift polygon. For example, the case $p/q = 3/7$ has been illustrated in Figure IV.4. The lines in Figure IV.5 connect

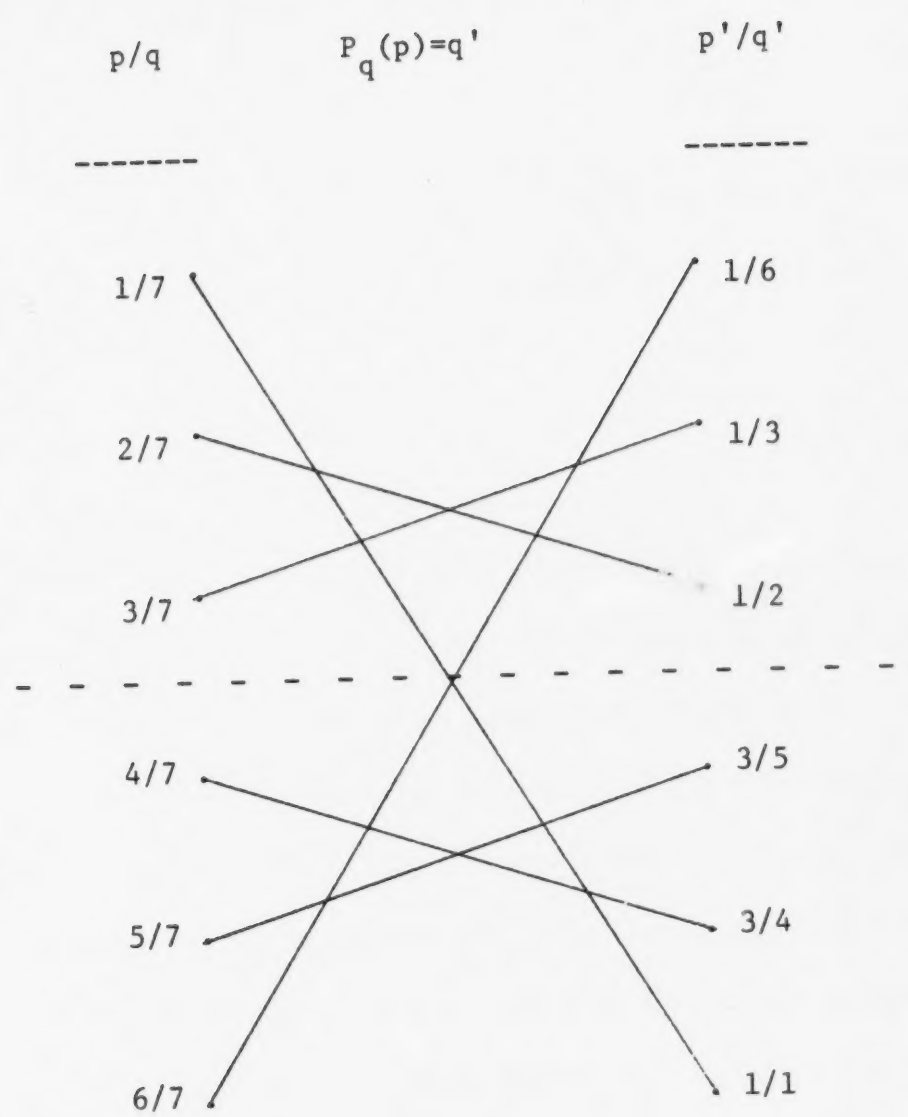


Figure IV.5: Function Relating Slopes with Fixed Denominators to Shift Polygons with Fixed Number of Vertices

fractions with equal values of p and q'. The right-left symmetry of this figure of lines is an expression of Theorem IV.12. That is, a value of p in the left column yields a value of q' opposite it in the right column; that value of q' in a numerator in the left column will then yield the value p in the denominator in the right column.

The vertical symmetry of the figure of lines in Figure IV.5 about the dotted line is expressed in the following.

Theorem IV.13: The shift polygon for (q-p)/q is the same as that for p/q with the directions of all chords reversed.

Proof: The transformation $C = \begin{pmatrix} 1 & 1 \\ 0 & -1 \end{pmatrix}$ discussed in III.1 maps (q, p) into (q, q-p) and (q', p') into (q', q'-p'). The cross product of the vectors from the origin to these new points is

$$(q'-p')q-(q-p)q' = qq'-p'q+pq'-qq' = -1.$$

Thus, from Theorem IV.11 (q', q-p) lies on the lower line of lattice points closest to the line of slope (q-p)/q through the origin. Therefore, downward translation of this line results in a right cyclic shift of the code for (q-p)/q by q' digits and upward translation results in a right cyclic shift by -q' = q-q' digits (i.e., a left cyclic shift by q' digits which is the same as a right cyclic shift by q-q' digits), whence

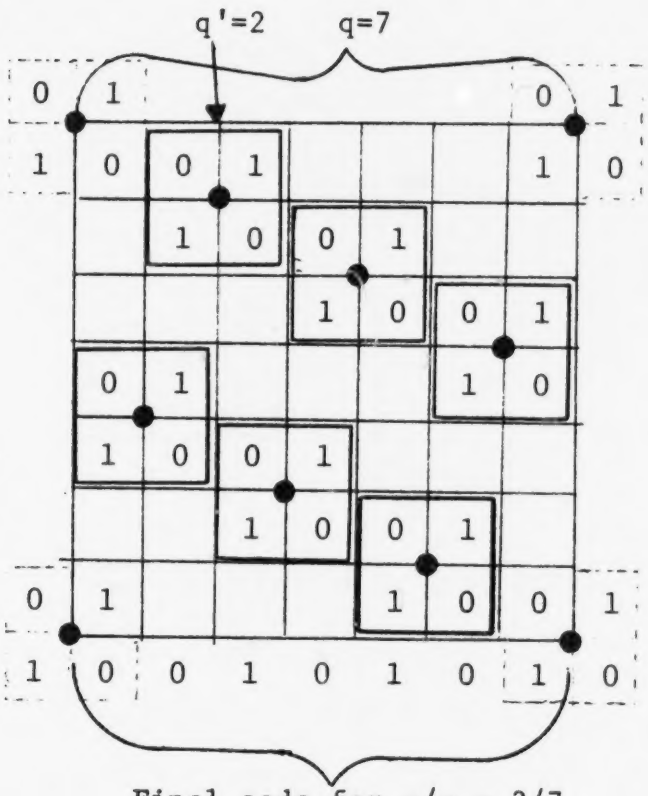
$$P_q(q-p) = q - P_q(p) .$$

The digit shift of a code of a line whose slope is a complement of another is the complement of the shift with respect to the denominator of that slope.

In the preceding paragraphs it was shown that codes for straight lines of rational slope have the property that transposing some adjacent pair of digits yields the original code cyclically shifted. If there are no periodic binary strings other than straight line codes that have this property, it can be used as the basis of straight line recognition.

Suppose there is some periodic binary string with period q containing 0 and 1 in columns k and k+1 which when transposed yield the original string cyclically shifted q' digits to the right where q and q' are mutually prime. Then this new string must have a 0 and 1 in columns k+q' and k+1+q' which when transposed yield the original string cyclically shifted 2q' digits to the right. Similarly, a sequence of such transpositions yields the original string cyclically shifted qq'mod(q) = 0 digits to the right, i.e., the original string. The following construction can be used to derive the original binary string from the sequence of transpositions.

Construct the transposition table for q and q' as follows. (Refer to Figure IV.6, the transposition table for q = 7, q' = 2.) The first row of cells represents the unknown code after the first transposition has occurred. The location of the first transposition determines the position of two digits of the code, i.e., the 1 and 0 resulting from the transposition. For convenience write this 0 in the leftmost cell of the top row and the 1 in the rightmost cell. Since the code is cyclic, the starting point is arbitrary. If such a transposition results in a cyclic shift by q' digits to the right in the code, the next transposition must occur q' digits to the right of the first. Using the second row



Final code for p/q = 3/7

Figure IV.6: Transposition Table for Cyclic Shifts

from the top of the grid to represent the unknown code after the second transposition, two more digits of the unknown code are determined by the location of this transposition, i.e., an adjacent 1 and 0 in the q'th and (q'+1)th columns. Similarly, after the jth transposition, a 1 and a 0 can be found in the jq' mod(q)th and (jq+1)mod(q)th columns of the jth row from the top of the table.

Since each transposition determines the columns in which an adjacent 0 and 1 occur in the jth row from the top of the table, it determines the location of the adjacent 0 and 1 in the row above corresponding to the code prior to that transposition. Call the grid point joining the four cells of the table holding these four digits a transposition point. If q' has no factors in common with q, there will be one transposition point between any two adjacent columns of the table after q transpositions. Therefore each digit of the code has taken part in two transpositions, one corresponding to the transposition point to the left and the other to the right of the column containing that digit. If the original digit in some column was 0, the first transposition replaces it with a 1 and the second replaces the 1 with a 0. If the original digit was 1, the first transposition replaces it with a 0 and the second replaces the 0 with a 1. Therefore, after q transpositions the original code results. It can be read off from the transposition table by taking in order either the bottom or the topmost entry in each column of the table.

It is impossible for q' to have any factors in common with q if a single transposition results in a string that is the original string

cyclically shifted. For if q' and q had the factor k in common, the transposition points would only lie on vertical lines to the right of columns that were multiples of k. Thus after q/k transpositions, a transposition point would recur between an adjacent 1 and 0, the results of the last time the transposition point occurred in that column. That is, the digits resulting from the first transposition in those columns would not be altered by transpositions in adjacent columns since none occurred. These digits are therefore in the wrong order for a transposition to occur and a contradiction results. That is, if the first transposition replaces 01 with 10, so must all others because the resulting code is identical to the prior code except for a cyclic shift.

It is clear from the construction just described that q and q' completely determine the structure of the transposition table and hence the code derived from it by reading off the bottom entry in each column. Now consider a straight line code of period q for which the transposition yields a cyclic shift by q' digits to the right. This straight line code will yield precisely the same transposition table as that of an unknown string with period q and right cyclic shift q' resulting from a single transposition. Therefore, the unknown string must be the code of the straight line with the same q and q'. This proves the following theorem.

Theorem IV.14: The codes of lines of rational slope are the only periodic binary strings containing an adjacent 0 and 1 which when transposed yield the original code cyclically shifted.

Since straight line codes correspond to configurations of excited cells on the lattice that can be crossed by straight lines, the theorem implies that only patterns which excite such configurations can yield regular shift polygons when translated in a fixed direction. That is, to the resolving power of the grid, the pattern must be a straight line.

IV.2.3 Lines as Operators on the Code

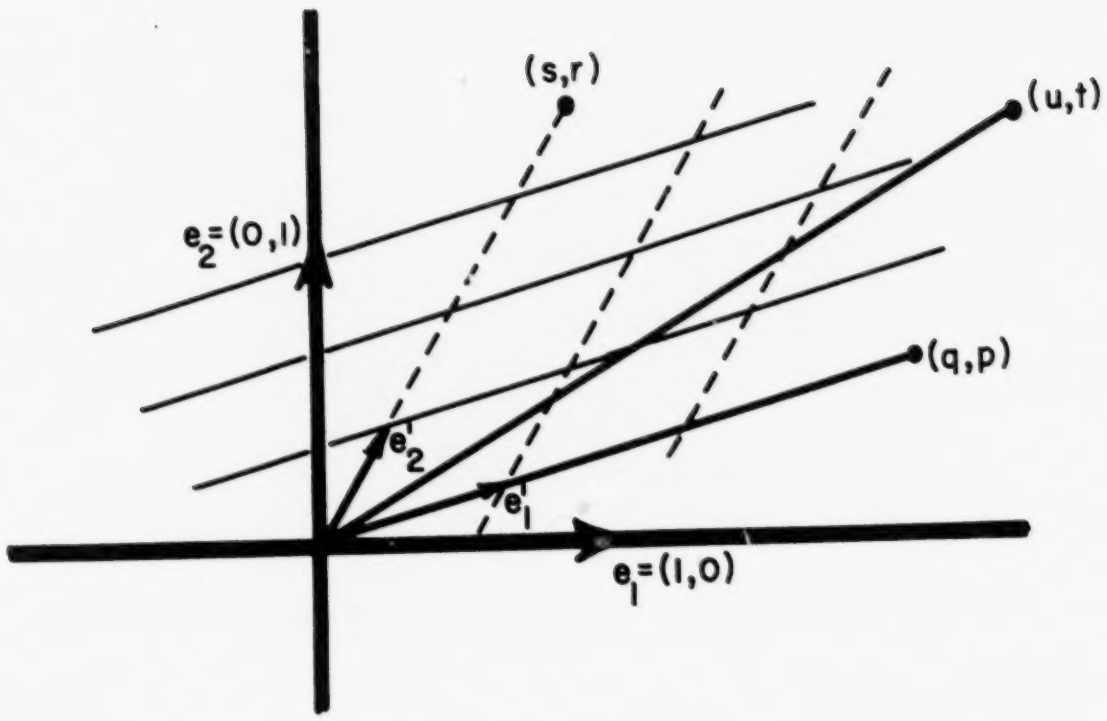
The following is an application of the code shift properties described in the preceding sections of IV.2. It is of interest because it shows how the trajectory of motion of a pattern, in this case a pair of lines, can be found relative to the grid without observing the change in coordinate values of points in the pattern induced by such motion. That is, the cyclic shifts in the lines caused by their translation on the grid can be used to find the magnitude and direction of that translation. Details follow.

Consider two lines L_1 and L_2 through the origin on an nxn grid. Let p/q and r/s be their respective slopes where $q \leq n$, $s \leq n$, and $0 < p/q < r/s < 1$. If the intersection of these lines is moved about the grid, preserving both slopes, the code of each will be cyclically shifted as described in the preceding sections of IV.2. Denote by c_1 the magnitude of the shift in the code of L_1 caused by displacing L_1 $(p^2+q^2)^{-1/2}$ units perpendicular to its direction. Denote by c_2 the magnitude of the shift in the code of L_2 caused by displacement of L_2 $(r^2+s^2)^{-1/2}$ units perpendicular to its direction. These shift magnitudes correspond to q' in preceding sections. Moving the intersection of L_1 and L_2 along a line of slope t/u, preserving their slopes, results

in shifts of both codes that depend on the components of motion perpendicular to L_1 and L_2. Conversely, given the codes of two lines and their sequences of cyclic shifts under translation it is possible to derive the trajectory of their intersection when translated along the line of slope t/u. In this sense, the line of slope t/u is said to be an operator on the codes of L_1 and L_2.

In Figure IV.7 the parallels to L_1 represent the displacements of L_1 which result in shifts of its code; similarly, the parallels to L_2 represent the displacements of L_2 which result in shifts of its code. Regarding these parallels as the horizontals and verticals, respectively, of an "operator plane", the code with respect to this new coordinate system of the line of slope t/u in the old system corresponds to the sequence of cyclic shifts in the codes for p/q and r/s as their intersection is moved along the line of slope t/u. That is, each 0 of the code represents crossing a "vertical" of the operator plane, causing a right shift of c_2 digits for the code of r/s. Similarly, each 1 of the code represents crossing a "horizontal" of the operator plane, causing a right shift of c_1 digits in the code for p/q.

To find the code of t/u in the operator plane the coordinates of (u, t) with respect to the lattice basis generating the operator plane are needed. Call the basis vectors of the original square lattice $\begin{pmatrix} e_1 \\ e_2 \end{pmatrix}$ and the basis vectors of the operator plane $\begin{pmatrix} e_1' \\ e_2' \end{pmatrix}$. The coordinates of the latter with respect to the former are found by simultaneously solving the equations of the lines whose intersections determine the fundamental parallelogram of the operator plane. For the coordinates of e_1' with respect to $\begin{pmatrix} e_1 \\ e_2 \end{pmatrix}$ solve:



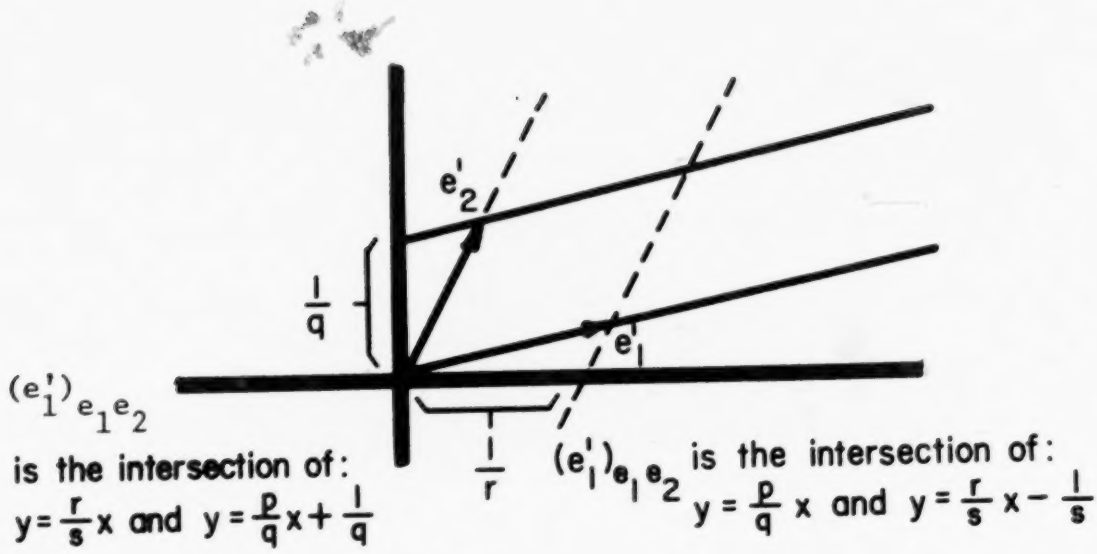


Figure IV.7: Operator Plane

$$y = (p/q)x \quad \text{and} \quad y = (r/s)x - 1/s$$

obtaining

$$e_1' = (q/(qr-ps), p/(qr-ps))_{e_1 e_2}$$

where the subscripts refer to the basis in which the coordinates preceding are to be taken. Similarly, for e_2' solve:

$$y = (r/s)x \quad \text{and} \quad y = (p/q)x + 1/q$$

obtaining

$$e_2' = (s/(qr-ps), r/(qr-ps))_{e_1 e_2}$$

These relationships are expressed by the matrix equation:

$$\begin{pmatrix} \frac{q}{qr-ps} & \frac{p}{qr-ps} \\ \frac{s}{qr-ps} & \frac{r}{qr-ps} \end{pmatrix} \cdot \begin{pmatrix} e_1 \\ e_2 \end{pmatrix} = \begin{pmatrix} e_1' \\ e_2' \end{pmatrix}$$

Now, to find the coordinates of $(u, t)_{e_1 e_2}$ with respect to the new basis postmultiply by the inverse of the above matrix:

$$(u, t)_{e_1 e_2} \cdot \begin{pmatrix} r & -p \\ -s & q \end{pmatrix} = (ru-st, qt-pu)_{e_1' e_2'}$$

Thus, the code for $(qt-pu)/(ru-st)$ gives the sequence of shifts for the codes of L_1 and L_2 as described in the preceding paragraph.

Now suppose p/q and r/s are known and the intersection of the lines L_1 and L_2 of those slopes through the origin is translated along a line of unknown slope t/u. For clarity, say $p/q < t/u < (p+r)/(q+s)$, i.e., the slope of t/u in the operator plane whose axes are L_1 and L_2 is between 0 and 1 so that an operator code can be defined without worrying about the octant reflections discussed in II.2. The value of t/u can

be derived from the sequence of cyclic shifts induced in the codes for L_1 and L_2 as follows. Every time a shift occurs in the code for L_2 write a 0 in the operator code. Every time a shift occurs in the code for L_1 write a 1 in the operator code and ignore the next shift in the code for L_2. The result is the code for t/u in the operator plane. That is, since the line through the origin and (u, t) on the grid corresponds to the line through the origin and (ru-st, qt-pu) in the operator plane according to the preceding paragraph, the operator code has (ru-st) digits and (qt-pu) of them are 1's. Calling the coordinates in the operator plane (u', t'), u and t can be found by solving for (u, t) in the matrix equation in the preceding paragraph,

$$(u, t) = (u', t') \begin{pmatrix} \frac{q}{qr-ps} & \frac{p}{qr-ps} \\ \frac{s}{qr-ps} & \frac{r}{qr-ps} \end{pmatrix}.$$

Considering the direction of motion of a pattern relative to the grid as described in the preceding paragraphs might be very useful in accurately determining the position of patterns on the grid relative to each other. That is, the unknown grid coordinates (u, t) of a cell can be found by moving the intersection of two known lines into that cell and observing the digit shifts in the codes of the two lines. Furthermore, since the choice of these two lines is arbitrary, they can be chosen in a way to set up an arbitrary temporary coordinate system on the grid. Motion parallel to one axis corresponds to an absence of shifts in the code of the line defining that axis and motion in other directions corresponds to a sequence of shifts in the codes of both lines. This

flexible coordinate mechanism could greatly simplify the interpretation of perspective change in patterns projected on the grid from a three dimensional space. That is, if a point in space has coordinates (u', t') with respect to the coordinate system of some plane in space in which it lies, various projections of that point and plane on the grid will result in a variety of cell excitation patterns. However, the coordinates of the projection of that point in the operator plane defined by the projection of the axes of the plane in space onto the grid will be constant.

The operator plane may be used to determine translation along trajectories other than straight lines. That is, translation of the intersection of L_1 and L_2 through any side of a cell defined by the intersection of operator plane parallels yields a shift in the code of one of the lines. This shift uniquely determines which side of the cell was crossed. Thus, an arbitrary path through cells can be uniquely determined from the sequence of shifts in the codes for L_1 and L_2. The resolution of this path is limited by the size of the cells, but their area is in general much smaller than the unit area of the original grid cells. For example, suppose p/q and r/s are chosen to be 1/n and (n-1)/n. Then the area of the cells in the operator plane is $1/(n^2-2n)$, the cross product of the numerators and denominators of the two fractions. This increased resolving power is a consequence of the properties of lines discussed in II.4.

IV.3 Rotation of Lines and Farey Series in Terms of Continued Fractions

The CCF structure of the code can be used to determine the changes in code resulting from rotation of a line about the origin as well as the translation described in the preceding section. According to II.5, rotating a line through the origin from slope 0 to 1 on an nxn grid yields a succession of codes of lines whose slopes correspond to the successive terms of the Farey series of order n, $\mathcal{F}_n$. Thus the rule for code changes caused by line rotation corresponds to the rule for finding successive terms of $\mathcal{F}_n$. If p/q is a term of $\mathcal{F}_n$, the preceding and following terms will be called p^-/q^- and p^+/q^+ respectively here. The latter term is called the successor of p/q in $\mathcal{F}_n$. The rule for finding the successor of p/q follows.

Rotating the line L in Figure IV.3 about the origin to increase its slope results in no change in its code until it crosses the rightmost lattice point on L'. That is, since there are no lattice points in the strip between L and L' no new cell sides are crossed until the first new lattice point $((k-1)q+q',(k-1)p+p')$ is crossed (k is the largest integer satisfying $(k-1)q+q' \leq n$). By the same arguments as those preceding Theorem IV.7, the code for the line from the origin to that point on L' is the concatenation of (k-1) periods of the code for p/q and one period of the code for p'/q'. By the argument following Theorem IV.9, these two codes correspond to 0's and 1's respectively in the (N+1)th level code. Therefore, the CCF of the code for p^+/q^+ must be

$$\lceil s_1, s_2, \ldots, s_N, k \rceil \ .$$

That is:

Theorem IV.15: The successor of p/q in the Farey series of order n is

$$\frac{kp_N - p_{N-1}}{kq_N - q_{N-1}} = \frac{kp - p_{N-1}}{kq - q_{N-1}}$$

where k is the greatest integer satisfying

$$kq - q_{N-1} \leq n.$$

Note that p'/q' is the special case where k=1. In case the CCF terms immediately preceding this 1 are all equal to 2, the number of CCF levels is drastically reduced. That is,

Theorem IV.16: $\overline{\lceil s_1, \ldots, s_j, 2, 2, 2, \ldots 2, 1 \rceil} = \overline{\lceil s_1, \ldots, s_j - 1 \rceil}$

Proof: Follows immediately from $\overline{\lceil s_1, \ldots s_k, 1 \rceil} = \overline{\lceil s_1, \ldots, s_k - 1 \rceil}$.

That is, a string of consecutive 2's in the CCF is annihilated by a following terminal 1, and the preceding term (s_j) is diminished by 1. There is no analogue of this telescoping property in CF convergents because the expressions for successive convergents are sums rather than differences. (See Theorems 149HW and IV.1).

Refering again to Figure IV.3, the preceding theorem reflects an interesting duality between a lattice point with mutually prime coordinates (q, p) and the line L' of slope p/q through the infinite number of lattice points closest to L. Each lattice point on L' corresponds to a distinct successor of p/q in a distinct Farey series. Now for each point (w, v) on L^o there is a distinct line through an infinite number of lattice points closest to y=(v/w)x. But Theorem IV.11 says that pw-vq = 1, thus implying that (q, p) lies on each of these lines. Therefore it lies on their intersection. So, corresponding to the infinite

set of lattice points on L' or L^o there is an infinite set of lines through (q, p), each of which goes through one lattice point on L' and one lattice point on L^o. To all lattice points on L^o except (q^o, p^o), (q, p) has the same relation as (q', p') has to (q, p). Figure IV.8 illustrates the intersection of the described lines at (q, p).

The preceding geometric duality corresponds to the CCF duality between

$$\lceil s_1, s_2, \ldots, s_N, \overline{k} \rceil \quad \text{and} \quad \lceil s_1, s_2, \ldots, s_N+1, \overbrace{2,2,2,\ldots 2}^{j \text{ terms}}, 1 \rceil$$

That is, the first CCF expresses the points of L' in terms of (q, p) and the second expresses (q, p) in terms of the lines through points on L^o. The corresponding code duality is the concatenation of periods of the code for p/q in the first case and breaking down the code for p/q into the codes for p'/q' and p^o/q^o in the second case. In the latter case the code for p/q has the same relation to the code consisting of the concatenation of one period of the code for p^o/q^o and j periods of the code for p/q as the code for p'/q' has to the code for p/q.

The CCF structure of the code yields some novel proofs of well known theorems on Farey series. For example,

Theorem 28HW: If p/q and p^+/q^+ are two successive terms of $\mathcal{F}_n$ then

$$p^+q - q^+p = 1.$$

Proof: $p/q = p_N/q_N$ and $p^+/q^+ = p_{N+1}/q_{N+1}$ by Theorem IV.15.

Thus, the cross product above is a case of Theorem IV.2.

Theorem 29HW: If p^-/q^-, p/q, p^+/q^+ are three successive terms of $\mathcal{F}_n$ then

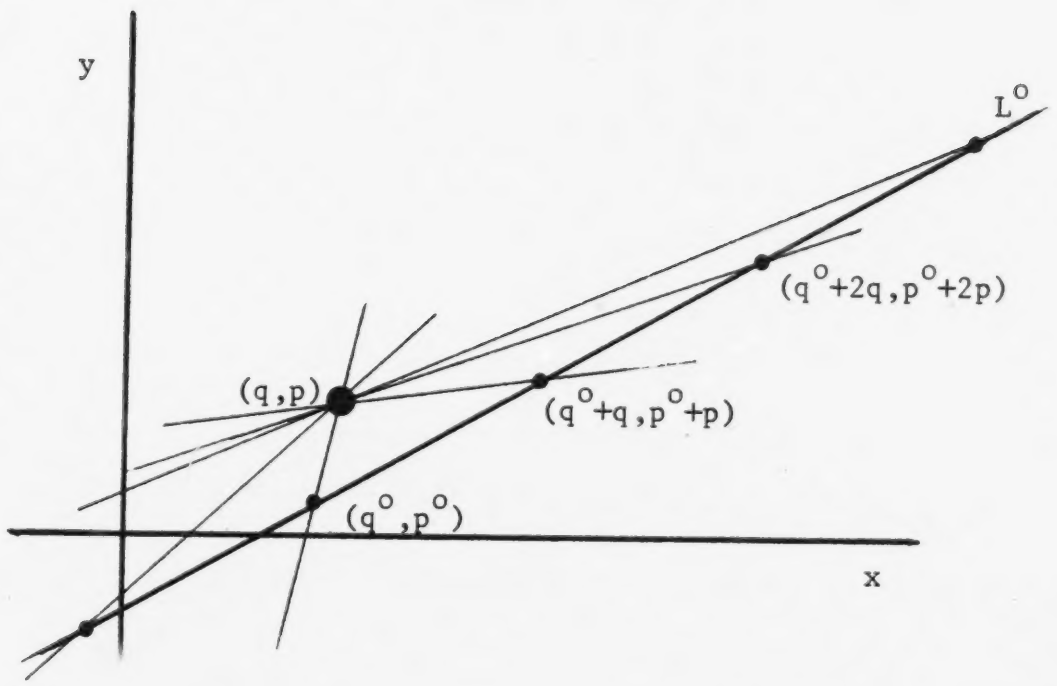


Figure IV.8: Geometric Representation of p/q and its Predecessors in all Farey Series'

$$\frac{p^+ + p^-}{q^+ + q^-} = \frac{p}{q} \quad .$$

Proof: The three fractions are three successive convergents to the CCF $\lceil s_1, s_2, \ldots s_{N-1}, s_N, s_{N+1} \rceil$.

Applying Theorem IV.1 yields

$$\frac{s_{N+1}p_N - p_{N-1} + p_{N-1}}{s_{N+1}p_N - q_{N-1} + q_{N-1}} = \frac{s_{N+1}\ p_N}{s_{N+1}\ q_N} = \frac{p}{q} \quad .$$

Notice that the preceding theorem holds for any three successive convergents to a CCF, not simply a CCF whose entries are determined by the order of the Farey series according to Theorem IV.15. A geometric interpretation of this property is that every CCF convergent lies on the main diagonal of the parallelogram formed by adding the vectors corresponding to the preceding the following CCF convergents. An analogue in CF convergents is:

Theorem IV.17: For the CF convergent v_i/w_i defined as in IV.1,

$$\frac{v_{i+1}-v_{i-1}}{w_{i+1}-w_{i-1}} = \frac{a_{i+1}v_i}{a_{i+1}w_i} = \frac{v_i}{w_i}$$

Proof: Applying Theorem 149HW yields

$$\frac{a_{i+1}v_i + v_{i-1}-v_{i-1}}{a_{i+1}w_i + w_{i-1}-w_{i-1}} = \frac{a_{i+1}v_i}{a_{i+1}w_i}$$

Geometrically, this means that the vector connecting the lattice points corresponding to any two successive odd/even CF convergents is a multiple of the vector from the origin to the point corresponding to the intervening even/odd CF convergent.

The new proofs of Farey series theorems based on the CCF and codes instead of traditional tools depend on the fact that the lattice point corresponding to the successor of any term p/q lies on the line L' of lattice points closest to the line L through the origin and (q, p). Under any unimodular transformation the images of the lines L and L' retain their property of being "closest" to each other (no lattice points from elsewhere in the original lattice are mapped onto or between them). Furthermore, points with mutually prime coordinates are mapped into points with mutually prime coordinates by unimodular transformations. These two properties make it possible to generalize the notion of Farey series as follows:

Definition IV.1 Let $\mathcal{F}_n$ be the usual definition of the Farey series of order n. Then let $\begin{pmatrix} x_1 & x_2 \\ y_1 & y_2 \end{pmatrix} \mathcal{F}_n$ where where $x_1y_2 - x_2y_1 = 1$ be called the Generalized Farey Series of order n, whose members are defined as follows. If the kth term of $\mathcal{F}_n$ is p/q then the kth term of the Generalized Farey series is

$$\frac{y_1q+y_2p}{x_1q+x_2p} .$$

The unimodular transformation above maps the unit square (0,0), (1,0), (1,1), (0,1) into the unit parallelogram (0,0), (x_1,y_1), (x_1+x_2,y_1+y_2), (x_2,y_2) and the square nxn grid into the parallelogram composed of nxn of these unit parallelograms. Thus for any unit parallelogram there are generalized Farey series of any order. Choosing a transformation to map the first quadrant into a wedge within the first quadrant introduces a refinement of the view of that subset of the quadrant. An

alternate interpretation is that the transformation is the discrete approximation of the affine perspective transformation of the two dimensional projection of three dimensional objects under rigid motion in three dimensions. The reduction sequences which lead to the continued fraction of the slope of a line may be interpreted in both the above ways. Each transformation in the sequence both refines the approximation to the slope and changes the perspective from which the line is viewed until the line is the x-axis of the final coordinate systems. The code is then reduced to the digit 0.

CHAPTER V: PARALLEL COMPUTERS WHICH RECOGNIZE AND GENERATE STRAIGHT LINES

The code theorems in the preceding chapters express, in the efficient language of the code, intrinsic geometric properties of lines on lattices. For example, Theorem II.2, which states that the number of 0's separating 1's is nearly the same throughout the code of a straight line, expresses the fact that a line is cut into equal segments by the parallels of the lattice. The code used here is particularly powerful in expressing such properties because geometric transformations of lines on lattices often correspond to simple code transformations.

In writing the code for a line, each digit is determined by the position of the line with respect to a single lattice square. Similarly, in CF- or CCF-reduction, each new digit is determined by a single segment of the old code. Hence, both these operations may be performed by parallel rather than sequential string processing. This suggests a parallel computer design to perform such operations. In this chapter the code properties described in preceding chapters are incorporated into the design of parallel computers that recognize and generate configurations of grid cells that can be crossed by straight lines.

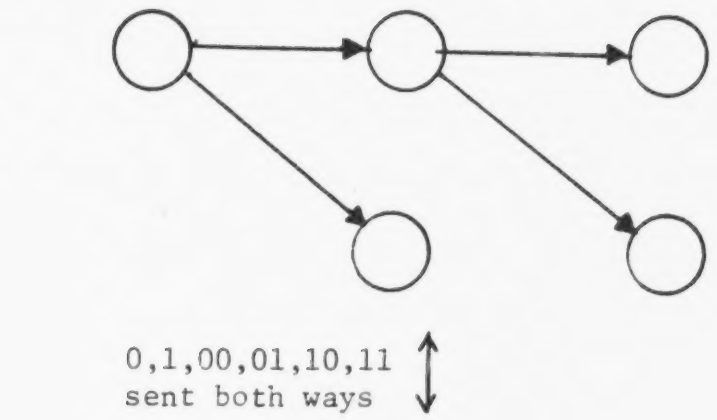
V.1 Structure and Notation of Parallel Computers

Each of the parallel computers to be described consists of a collection of identical finite automata operating in parallel. Their programs and inputs vary according to the task. The structure, notation, and mode of operation common to all the parallel computers is described

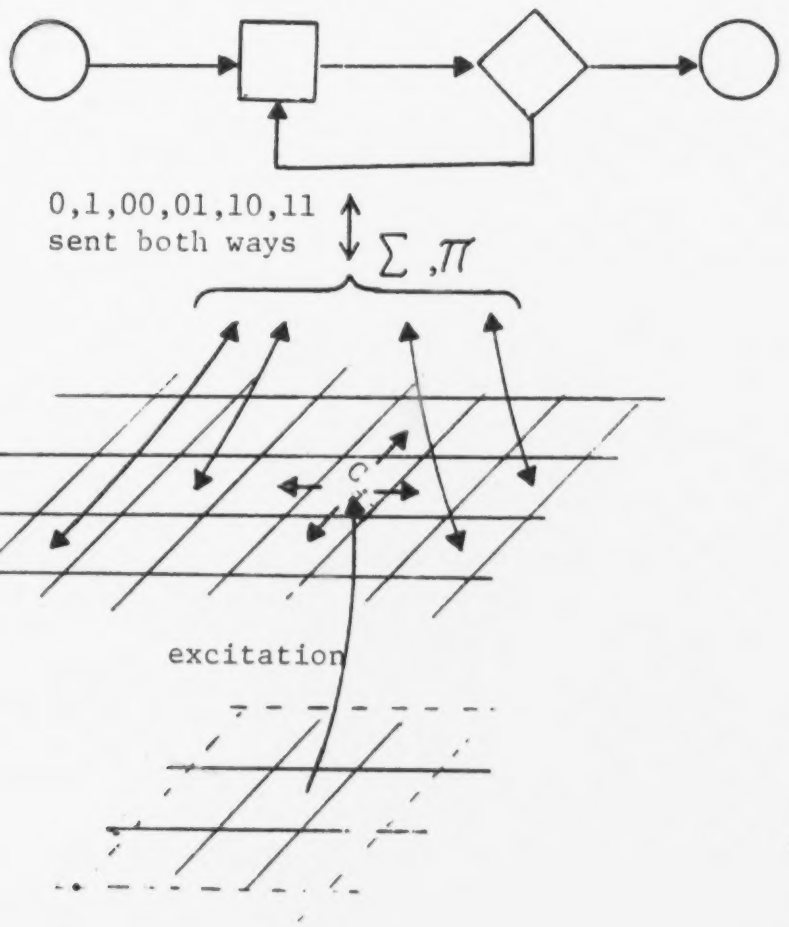
briefly in the following paragraphs. Details will be given in the descriptions of specific designs in the following sections.

Each parallel computer consists of sets of automata in four levels, each of which performs a different kind of informational task and communicates with the levels immediately above and below. Messages sent from higher to lower levels may be regarded as instructions to the lower level to 'look for' something and messages in the opposite direction as reports that something has or has not been found. At the lowest level are the external input sensors, e.g., photocells, which detect the presence or absence of a stimulus, e.g., light, in a single grid square and send a 1 or a 0 to the automaton in the next higher level (see Figure V.1). Each automaton, called a C (Cell) in the latter level is thus associated with a single grid square. Each C exchanges messages with the C's that are its four nearest neighbors. These messages consist of either 0, 1, or no message, sent independently to all four neighbors. In addition, C's exchange one of the messages (0, 1, 00, 01, 10, 11) with an automaton called the CS (Cell Synchronizer) in the next higher level.

The CS connects to a large number of C's but is not capable of distinguishing the identity of individual sources or recipients of its inputs and outputs. It combines the messages from the C's in a symmetri boolean function and according to the value of that function goes into a new state. At that time, it sends the same message to all C's connected to it. The CS therefore, does not store information



CS^2



, CS

C

stimulus detector

Figure V.1: Organizational Hierarchy of Parallel Computers

from individual cells nor perform computations. It merely synchronizes and programs the C's so that they are performing the right operation at the right time.

The general mode of operation and notation used to express it are as follows. First, the pattern is projected onto the stimulus detector level and detectors that are excited by it send 1 to the C's which respond by going into an active state and connecting to CS. The states of C's are represented by four digit binary numbers in the ovals in the transition graph (e.g., left half of Figure V.2). The states are written as four digits to permit the storing of different kinds of information in different digits; the inactive state prior to excitation is 0000. All automata in the parallel computer are Mealy type; i.e., there is a finite delay between an input and the resulting output to neighbors to prevent race conditions. Upon excitation, all C's simultaneously send outputs to their four nearest neighbors and CS. Messages between C's are written in the corresponding quadrants of the large X's adjacent to the appropriate transition arrow in the left half of Figure V.2. Inputs to a C are written above the horizontal line next to this arrow and outputs below. For example, written above the line means a C receives a 1 from both its upper and left neighbors. Written below the line, the same symbol means the C is sending a 1 to those neighbors. If a given C transition does not depend on inputs from other C's but only on the input from CS (a "don't care" condition for input from C's) the X will be ommitted. If the transition condition is a single symbol e.g., 1, from any single neighbor, that symbol will be written without

an X. (For example, see the transition from 1100 to 1110 in Figure V.3.) The C messages to and from CS are written in the samll squares below and above the same horizontal lines in the left half of Figure V.2.

In the time step following the C transition described in the preceding paragraph, CS receives simultaneously the messages from all C's connected to it and computes some simple symmetric boolean function of them. The value of this function and the present state of the CS determine its next state and output. The states of the CS are written in the circles in the right half of Figure V.2 and the inputs and outputs are written above and below the horizontal line adjacent to the appropriate transition arrows. In writing the boolean functions, the variable z_{ij} refers to the one digit message from the ijth C to CS and is either 0 or 1; x_{ij} and y_{ij} refer to the first and second digits of two digit messages from the ijth C to CS. The common notation $\prod_{ij}$ and $\sum_{ij}$ are used for boolean product and sum respectively. The subscripts will be omitted in the transition graphs since they are always the same. The output from CS to the C's is written in the small square below the horizontal line.

In the next time step the C's receive inputs simultaneously from neighbors and CS. The process just described repeats so that activity alternates between the C and CS level of the parallel computer. When a recognition process is complete, a message may be sent from CS to CS^2, an automaton in the highest level that may respond to and/or direct the operations of several independent CS's. The transition graph for a CS^2 is at the bottom of Figure V.2.

Messages to and from CS^2's are written in diamond shaped boxes next to the appropriate transition arrows. The details of notation and operation will become clearer in the discussion of specific examples that follow.

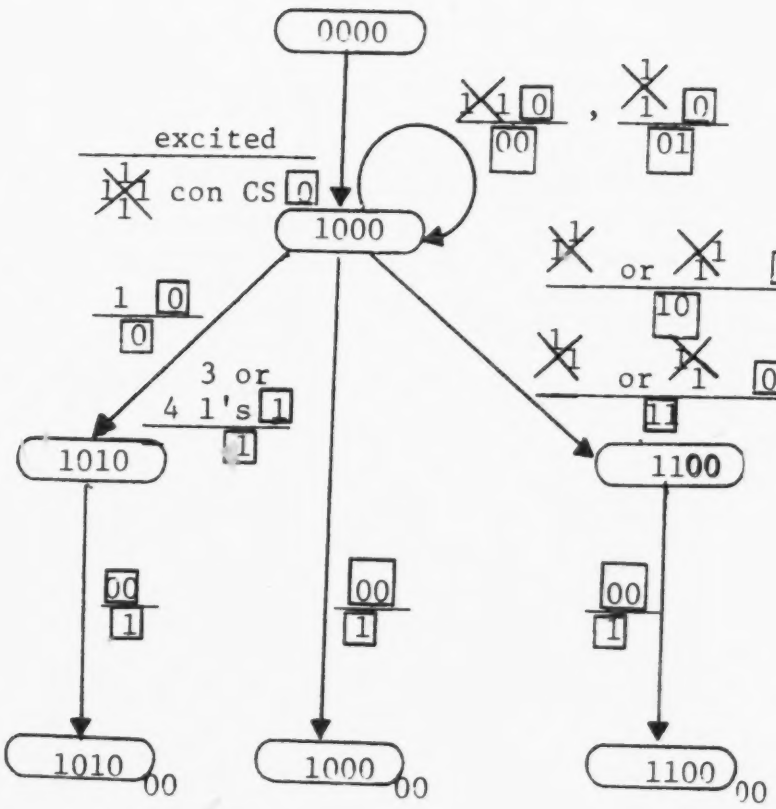
V.2 Straight Line Recognizer Based on CCF-Reduction

This design implements the CCF-reduction (described in Theorem III.4) on the code of a configuration of excited C's on the grid. It is based on the simultaneous deletion of 0's from code segments followed by the conversion of 1's terminating long segments to 0's and those terminating short segments to 1's.

The general sequence of operations is as follows. First, C's excited by the pattern go into states corresponding to code digits based on the locations of their nearest excited neighbors. C's corresponding to 1's then send messages to one of their neighbors. This message propagates from each C corresponding to a code 1 along adjacent C's corresponding to code 0's, converting the latter to a state in which they do not correspond to any code digit. When all such C's have been "erased" in this manner, the remaining C's, which correspond to code 1's, are converted into states corresponding to 0's or 1's depending on the number of steps it took for the message propagating along C's corresponding to 0's to reach them, i.e., depending on whether the 1's of the original code terminate long or short segments. If this number of steps differs by more than one between any two adjacent 1's, the pattern is rejected since Theorem II.2 excludes from line codes those

strings in which segment lengths differ by more than one digit. If this condition is not violated, the propagation of messages from states corresponding to 1's starts over again and repeats until only the code digit 1 is represented in the states of C's. Recognition is then complete. Details follow.

Figure V.2 illustrates the determination of code 0's and 1's and the test for octant compatibility of the corresponding nearest neighbor configurations. Initially all C's are in state 0000, and CS and CS^2 are both in state 0. Then, all C's excited by the pattern go into state 1000 send 1 to each of their four nearest neighbors and connect to CS, sending the latter 0. All the events in the preceding sentence occur simultaneously. CS, which received 0's from all C's, responds in the next time step by staying in state 0 and sending 0 to all C's simultaneously. This is interpreted by each C as an instruction to determine the configuration of excited nearest neighbors based on the direction from which 1's arrive that were sent by all C's to all their nearest neighbors in the previous time step. That is, C's receiving from right and left neighbors or upper and lower neighbors stay in state 1000 and send 00 or 01 respectively to CS. C's receiving from asymmetrically placed neighbors, on the other hand, go into state 1100 and send 10 or 11 to CS. Thus, the first digit of the C state holds the information that the C is excited by the pattern and the second holds the straight line code digit corresponding to the local neighborhood configuration of excited C's. The first digit of the two digit messages sent to CS at this time tells whether the configuration is a



$$\frac{\sum_{z=0}}{0} \quad (a)$$

$$\frac{\sum_{z} + \sum_{xy}\sum_{\bar{x}\bar{y}} + \sum_{\bar{x}y}\sum_{x\bar{y}} = 1}{\text{stop}} \quad (b)$$

$$\frac{\prod_{xy} + \prod_{\bar{x}\bar{y}} + \prod_{\bar{x}y} + \prod_{x\bar{y}} = 1}{\text{stop}} \quad (c)$$

$$\frac{\sum_{\bar{x}\bar{y}}\sum_{x\bar{y}} = 1}{00} \quad (d)$$

$$\frac{\sum_{\bar{x}y}\sum_{x\bar{y}} = 1}{10} \quad (e)$$

$$\frac{\sum_{\bar{x}y}\sum_{xy} = 1}{11} \quad (f)$$

$$\frac{\sum_{\bar{x}\bar{y}}\sum_{xy} = 1}{01} \quad (g)$$

Figure V.2: Octant Compatibility Detector

run or bend, i.e., it is the corresponding code digit. The second digit gives the orientation of the run or bend. Figure II.2 illustrated these run and bend configurations during the initial discussion of code octant symmetries.

The preceding paragraph described only C's receiving 1's from two nearest excited neighbors. Those receiving from one, three, or four neighbors go into state 1010. If the chain of excited C's corresponds to a straight line, those C's receiving from one excited nearest neighbor must be on the ends of the chain; their role in the recognition process will be described later. If any C's receive 1's from three or four nearest excited neighbors, the pattern cannot correspond to a straight line; such C's signal their presence by sending 1 to CS.

CS now examines the messages sent to it by the C's to determine whether there are conflicts between the octants compatible with the nearest neighbor configurations of excited C's. That is, if CS simultaneously receives 00 and 01 or 10 and 11, there must be two types of run or two types of bends among the C's connected to the CS. Therefore, from the discussion in II.2, the pattern cannot be a straight line. Also, if there are any C's with three or four excited nearest neighbors, signaled by the arrival of a 1 at the CS from the C's, the pattern cannot be a straight line. These two kinds of incompatibility with straight line configurations are expressed by the boolean function in transition condition (b) in Figure V.2. In this case, CS sends 0 to CS^2 and halts. CS^2 responds by going into state 1, rejecting the pattern.

Now if all local configurations are bends or all are runs, no further processing is needed because the cell excitation pattern must be that of a straight line. The boolean function in transition condition (c) of Figure V.2 expresses this case, to which CS responds by sending 1 to CS^2 and halting. CS^2 responds by going into state 2, accepting the pattern as a straight line.

If none of the conditions (a), (b) or (c) are met the pattern must consist of runs of a single type and bends of a single type. The pattern therefore corresponds to a string of code satisfying the octant constraints but not necessarily the digit distribution constraints of straight lines. Transition conditions (d), (e), (f), and (g) correspond to the response of CS to the messages from C's indicating octants I, II, III, IV respectively. The first digit of the two digit message sent from CS to the C's at this time is the run type and the second is the bend type of the octant compatible with all the messages sent to CS in the previous time step (Figure II.2 illustrates these run and bend types). C's respond to this message by going into a state with the same four digit label but subscripted by the message from CS. The subscript determines the direction in which messages will be sent between C's during code reduction. Only the case for message 00, corresponding to octant I, is shown because in the other cases reduction will be carried out in exactly the same manner, differing only in the direction messages are sent.

Figure V.3 illustrates the CCF reduction for configurations of cells compatible with octant I. The transition graphs for other

octants are derived by replacing the subscripts with those corresponding to the run and bend types compatible with other octants and reflecting all the large X's containing input or output messages to and from C's about the corresponding axes or diagonals. For example, the reduction of a code corresponding to a line in octant III, is derived by using the subscripts 11 and reflecting the large X's containing inputs and outputs about the lines x=y and then x=0. This is easily understood by looking at Figure II.2.

Referring now to Figure V.3, C's in state 1000 correspond to runs, and those in state 1100 correspond to bends. Thus, tracing the chain of excited C's between the C's on the end in state 1010 yields the binary code of definition II.1 except that bend pairs correspond to two digits instead of one. This duplication is eliminated by the 01 sent from CS to C's in response to the 1's CS received from C's in the previous step. That is, C's in state 1100, corresponding to bends, send 1 to their upper neighbors and 01 back to CS. CS, now in state 1, responds by sending 00 back to the C's which is interpreted by bends as an instruction to look for a 1 sent by their neighbors in the previous time step. Those receiving such a 1, i.e., the upper member of each bend pair, go from state 1100 to state 1001 and disconnect from CS. Any C whose rightmost state digit is 1 is in a "conducting" state. That is, it conducts messages it receives from one neighbor to its other neighbor, effectively making these neighbors adjacent to each other. C's in state 1000 receiving 00 from CS remain in that state and send 0 to CS.

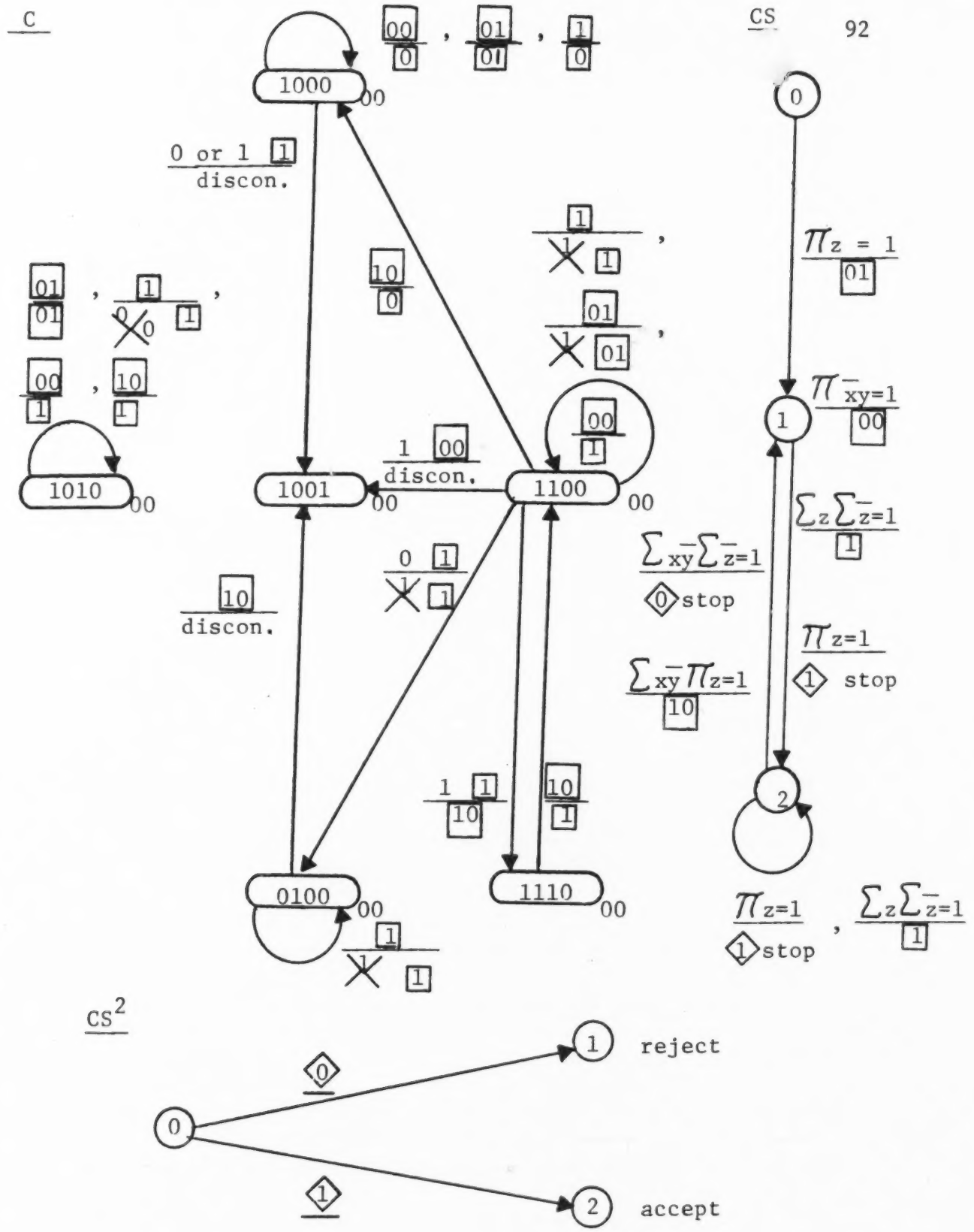


Figure V.3: Line Recognizer Based on CCF-reduction

CCF-reduction can now begin. CS, which received a mixture of 0's and 1's from the C's sends them all 1 while going into state 2. C's in state 1100, corresponding to code 1's, respond by sending 1 to their upper neighbors and 1 back to CS. The 1 sent to the upper neighbor is conducted through it to the first C corresponding to a code 0. Such C's, in state 1000, respond to the 1 from a neighbor and 1 from CS by going into the conducting state 1001 and disconnecting from CS. In this way a single code 0 is removed simultaneously from each code segment. C's in state 1000 that do not receive 1 from a neighbor send 0 back to CS. As long as CS receives a mixture of 0's and 1's from the C's, indicating there are still some code 0's left, it continues to send 1's to the C's, instructing them to continue deleting equal numbers of 0's from each code segment.

If the lengths of code segments only differ by one digit the last remaining C's corresponding to code 0's will be deleted at the same time as some C corresponding to a code 1 is receiving a 1 from a neighbor for the first time. The latter event is represented in Figure V.3 as a C in state 1100 going into state 1110 and sending 10 to CS. Since the last remaining code 0 has been deleted, CS receives a mixture of 1's and 10's, but no 0's. It responds by going into state 1 and sending 10 to the C's. This causes the C's in state 1110, corresponding to code 1's that received 1 from a neighbor and lie at the end of short segments of the code, to go into state 1100. C's in state 1100, on the other hand, correspond to code 1's at the end of long segments since they did not receive 1's from neighbors. Therefore, they respond by sending 0 to

to CS and going into state 1000, corresponding to code 0's.

One step of the CCF-reduction has now been carried out; long segments have been replaced by 0's and short segments by 1's. CS, now in state 1, receives a mixture of 0's and 1's and responds by going into state 2, sending a 1 to all C's and thereby commencing another CCF-reduction step by deleting a single 0 from each code segment as described in the preceding paragraph. If the CS ever receives no messages from code 0's while it is in state 1, reduction is complete so CS sends 1 to CS^2 and stops. That is, there are no more code 0's. CS^2 goes into s tate 2, accepting the pattern as a straight line. The number of times CS entered state 1 is the number of levels in the CCF expansion of the slope of that line, and the number of times CS entered state 2 between two successive returns to state 1 is the length of the long segment at that level, i.e., the CCF coefficient corresponding to that level. Therefore, the number of steps necessary for recognition is the sum of the coefficients plus the number of levels in the CCF expansion of the slope of the line. Thus, although there is a sequence of steps, the time (number of steps) required for recognition does not increase with grid size or number of excited C's. Instead, it goes up with the complexity of the continued fraction and the number of steps in general is far less than the number required by sequentially processing a chain of

If segment lengths of the code of the pattern differ by more than one digit at any level, CS will receive both a 10 and a 0 while in s tate 2, indicating that there are still code 0's that have not been deleted when the message propagating from a bend arrives at the end of

a short segment. This violation of Theorem II.2 causes CS to go into state 1, send CS^2 a 0 and stop. CS^2 responds by going into state 1 and rejecting the pattern as not being a straight line.

Since the preceding recognition process takes place on a finite grid, the C's terminating the chain of excited C's must be accounted for during the reduction. Recall that such C's have only one excited nearest neighbor . In particular, since any of these terminating C's could represent either a 0 or a 1 of the code that is truncated by the boundary of the grid or the end of a line segment, their behavior must be compatible with both possibilities. These C's, in state 1010, respond to the 1 from CS by sending 0 to their upper and right neighbors. C's in state 1000, corresponding to code 0's, respond to this 0 from a neighbor during CCF reduction by going into state 1001 and disconnecting from CS. That is, the terminating C is treated as a code 1 by C's corresponding to code 0's. However when a C in state 1100, corresponding to a code 1 receives a 0 from a neighbor it goes into state 0100 and continues to act like a code 1, sending 1 to its upper neighbor. During continued deletion of code 0's, caused by the continued sending of 1's from CS to the C's, this C in state 0100 acts as a barrier to the 0's it receives from its left. When the 10 comes from CS, causing digits to be rewritten according to the CCF algorithm, the CS in state 0100 goes into state 1001, eliminating the barrier to 0's coming from the C in state 1010 terminating the left end of the chain of excited C's. Since there are now only conducting C's between it at the C in state 1010 terminating the left end of the chain, this C formerly in

state 0100 acts as a chain terminating C to its upper neighbor during the next CCF reduction step.

Other designs for straight line recognizers are possible based on the CF reduction or the reduction of Theorem III.1. The transition diagrams representing such parallel computers need not differ greatly from the design just described, for many of the operations common to them all could be expressed by the same kinds of transitions. For example, deleting or rewriting digits, two operations common to all reductions, can be expressed as transitions into a conducting or non-conducting state, respectively, in the left half of all the transition graphs. Regardless of the particular reduction chosen and the transition graph chosen to represent that reduction, it is important to note that the structure and program of the individual C's and the CS unit overseeing their operation is fixed regardless of grid size or the size and shape of the pattern projected onto the grid. The only things that change when either grid size or pattern change are the number and location of C's connected to the CS and the sequence of transitions in the graphs of both kinds of units. Since the CS does not receive information about the number and location of C's connected to it, changes in such information do not effect the CS.

Following is a description of another kind of parallel computer that recognizes straight lines. It is not based on code reduction but on the code shift properties of a line that is translated on the grid.

V.3 Straight Line Recognizer Based on Shift Polygon

This design is based on the cyclic shift properties of straight line codes under translation, described in IV.2. In this case the pattern is moved relative to the grid and the resulting change in the configuration of excited C's is used to determine whether the pattern is a straight line. This mode of operation is markedly different from all other parallel computers discussed in Chapters V and VI, which analyze fixed patterns. It shows that pattern motion relative to the grid (such as image motion relative to the retina during eye tremor or drift) can be useful rather than detrimental in pattern recognition.

Recall that translating a straight line of slope p/q upwards 1/q units results in the transposition of a 0 and 1 in each period of the code, and this transposition yields the same result as a cyclic shift. A succession of such translations thus yields a succession of the corresponding cyclic shifts. The original code recurs after upward translation by one unit, which corresponds to q cyclic shifts, each of the same magnitude. According to Theorem IV.14, a straight line code is the only binary string with this property. It is recognized by the parallel computer as follows. Each C that is recently excited due to the motion of the pattern sends a message in both directions along the chain. If the pattern corresponds to a straight line, this message will arrive at C's previously and more recently excited in the same number of steps in all cases. That is, if the sequence of excitation of C's due to pattern motion constitutes a palindrome in space (number of intervening C's) and time (order in which C's are excited) the pattern is

accepted as straight line.

In order to see if a pattern satisfies the cyclic shift property of straight lines outlined in the preceding paragraph, the projection of the pattern on the grid must yield a configuration of excited C's for which a code can be written. That is, the C's must have run and bend configurations compatible with a single octant (see Figure II.2 and surrounding text) and no C's can have more than two excited nearest neighbors. In the preceding discussion of the line recognizer based on the CCF reduction (section V.2) the above criteria were tested by the parallel computer whose transition graph was given Figure V.2. The same program is to be applied here. That is, the pattern is to be rejected if it fails to satisfy the octant compatibility and number of nearest neighbors criteria, and accepted as input for the straight line recognizer otherwise. A complete discussion of that process was given in section V.2 and will not be duplicated here. After the processing corresponding to the transition graph in Figure V.2, all excited C's go into state 0000, subscripted by the appropriate run and bend type digits and further interactions are to be accounted for by the transition graph in Figure V.4 whose description follows. The transition graphs for other octants are derived by the same reflection of nearest neighbor specification and subscript substitution as described for Figure V.3.

Refer now to Figure V.4 Initially all C's are in state 0000 and CS is in state 0. Excited C's go into state 1010, send 00 to CS and 0 to their upper neighbors. CS then goes into state 1, sending 0 to the

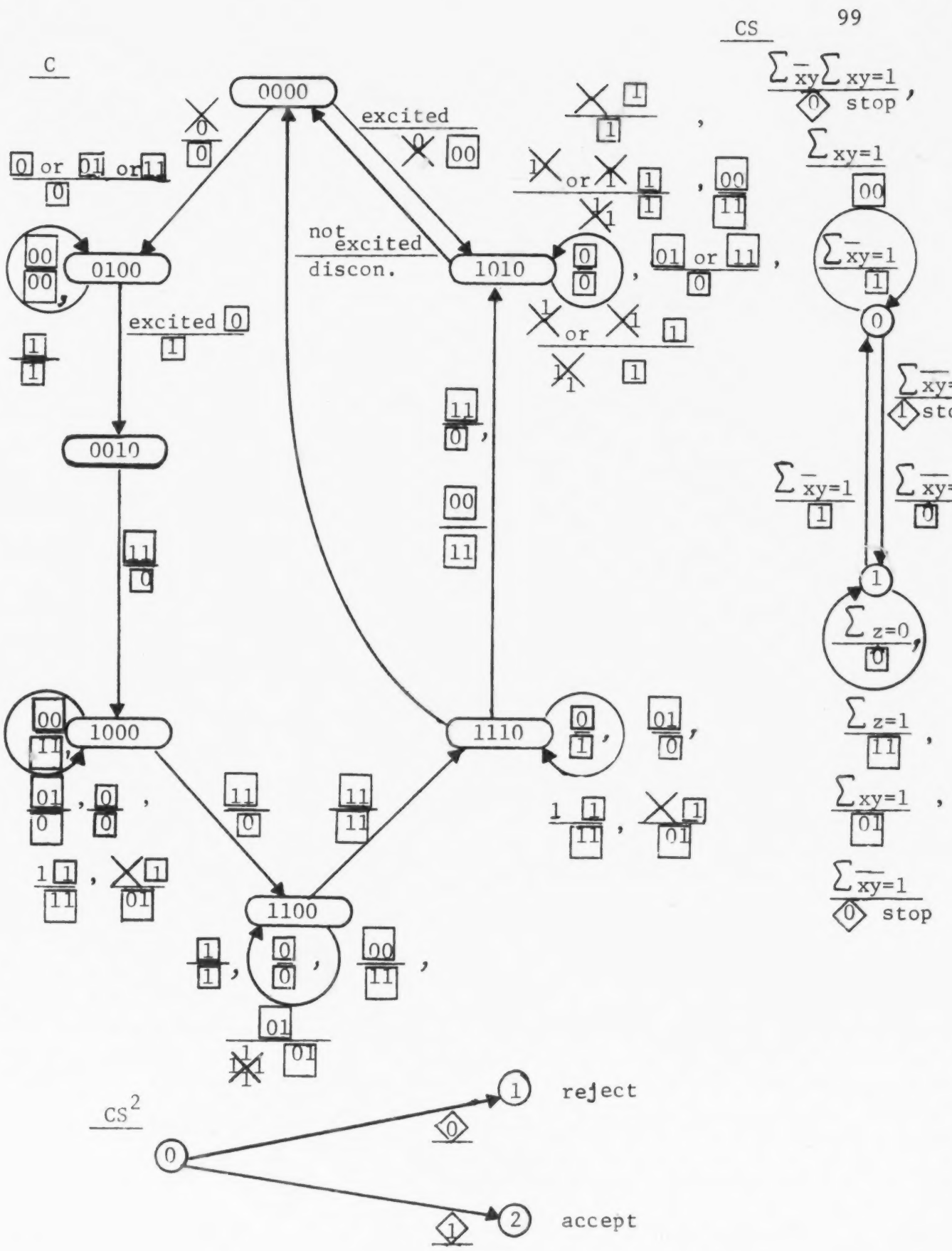


Figure V.4: Straight Line Recognizer Based on Shift Polygon

C's. C's in state 0000 receiving 0 from a lower neighbor go into state 0100 and send 0 to CS. Since these latter C's are directly above the C's excited by the pattern, they have the same configuration as the excited C's. Thus they can be used later to see if after q shifts the original configuration results. CS, now in state 1, receives 0's from all C's and stays in state 1. This exchange of 0's between CS and C's repeats indefinitely until the line is moved upward enough to excite one of the C's in state 0100 which then goes into state 0010 and sends 1 to CS. This new excitation corresponds to a digit shift in the code. CS responds by sending 11 to the C's causing those in state 0010 to go into state 1000 and send 0 to CS. C's in state 1000, 1100, and 1110 represent those most recently excited, second most recently excited and third most recently excited respectively, of those excited by the pattern as it moves upward (i.e., the number of 1's in the state is the index of "recentness of excitation"). The way the computer checks for equal spacing between successive shifts is to send a signal forward and backward along the chain of excited C's, starting at the second most recently excited C (in state 1100). If the most recently excited and third most recently excited C receive this signal simultaneously the spacing from first to second is the same as from second to third. Obviously this process cannot begin until three successive sets of C's formerly in state 0100 have been excited. Thus CS, on receiving all 0's, indicating that none of the C's have gone into state 1110, stays in state 1 and sends 0 to the C's, which in turn send 0 back. The loop is broken when a new C in state 0100 becomes excited and goes into

state 0010 as the line is moved upward. This triggers CS to send 11 to the C's, converting C's in state 1000 to state 1100. The third time a set of C's in state 0100 is excited, CS again responds with a 11 to the C's. Now there is a set of C's in state 1100 which respond to this 11 from CS by going into state 1110 and sending 11 to CS indicating that there are now three successive sets of newly excited C's.

CS, in state 1, now begins comparing the number of C's between digit shifts (corresponding to newly excited C's) by sending 01 to the C's. This causes the C's in state 1100 to send 1 to all four nearest neighbors and 01 to CS. On receiving 01, CS goes into state 0 and sends 1 to the C's, signalling them to pass the signal on. Those C's in state 1010 receiving 1 from a lower or left neighbor and 1 from CS send 1 to their upper and right neighbors and 1 to CS. Similarly, C's in state 1010 receiving 1 from an upper or right neighbor send 1 to a lower or left neighbor. C's in states 1000 and 1110 receiving 1 from CS and nothing from neighbors send 01 to CS, causing it to stay in state 0 and send 1 to the C's. Thus, these 1's are propagated through the excited C's until they reach a C in state 1000 or 1110. If any C's in those two states are not receiving relayed 1's at the same time that others are, the number of C's between successive digit shifts is not equal and CS responds by staying in state 0, sending 0 to CS^2 and stopping, thereby rejecting the pattern. That is, CS^2 goes into state 1. On the other hand, if all C's in state 1000 and 1110 do receive relayed 1's from neighbors simultaneously, they send 11 to CS, which then sends 00 to the C's. If there are still some unexcited C's in

state 0100 they send 00 to CS which then goes into state 1 and sends 0 to the C's. When C's in state 1110 receive 00 from CS they go into state 1010 and send 11 to CS. Thus there are now no C's in state 1110 left and CS receives 0's from all C's in response to its 0. This exchange of 0's between CS and C's continues as in the very beginning until a new C in state 0100 is excited by the upward moving line, triggering the same sequence of events as described previously. If a C in state 0000 is excited by the motion of the pattern, it sends 00 to CS and goes into state 1010. CS responds by sending 0 to CS^2 which then goes into state 1 and stops, rejecting the pattern.

There are two ways the parallel computer can stop. CS^2 halts in state 2 if the pattern is accepted as a line and halts in state 1 it it is rejected. The former is occasioned by the completion of relaying 1's from a C in state 1100 and the absence of any more unexcited C's in state 0100. This occurs when CS is in state 0 and receives no 00 messages from the C's, indicating that all digit shifts were of equal magnitude and the configuration is now the same as the original pattern. This results in a 1 being sent to CS^2 which halts in state 2, accepting the pattern as a straight line. Rejection was described in the preceding paragraph.

C's with only one excited nearest neighbor are not accounted for in this design because the related transitions would obscure the cyclic character of the transition graph (Figure V.4) that corresponds to the cyclic shift properties of the code. This discussion is nevertheless complete for a line on an infinite lattice or on a finite grid in which

columns that are separated by some multiple of q columns are identified and rows that are separated by the same multiple of q are also identified. Then the finite grid has the connectivity of a torus, and the chain of excited C's corresponding to a line is closed on that surface. In contexts in which it is desirable to include C's with only one excited nearest neighbor, e.g., recognition of a line segment on a grid, the related transitions can be accounted for as in section V.2.

This kind of line recognizer could be used in the context of lines as operators (described in IV.2.3). That is, each CS transition from state 1 to state 0 is a response to a C that is freshly excited by motion of the line, corresponding to a single cyclic shift in its code. If the slope of the line is p/q, such a shift is caused by translation of the line relative to the grid whose component perpendicular to the line is $(p^2+q^2)^{-1/2}$. If two lines with unequal slopes are connected to two CS's of this type, the magnitude and direction of motion of the pair of lines relative to the grid can be determined from the sequence of transitions of these CS's. That is, the two components of motion perpendicular to the two lines uniquely determine that motion.

Notice that although this recognizer is based on the same line code as the recognizer described in V.2, it uses a totally different property of the code as the basis for recognition. In fact, the digits of the code are not even represented in the states of C's during recognition by the parallel computer based on the shift polygon. The timing and spacing of changes in excitation of C's is used instead of the configuration of local neighborhoods of excitation. The magnitude

of motion necessary for recognition is the unit cell, no matter what the slope of the line. This kind of operation reminds one of a kind of mechanism available to live retinas because it depends on the sequence and timing of on-off type events during motion of a pattern relative to the detector array and on a simple distance comparison scheme that could be implemented by neighborhood interactions (such as lateral inhibition).

V.4 Straight Line Generator

In Euclidean geometry, two points determine a unique line. The parallel computer described here determines the grid cells that would be crossed by a straight line connecting the lower right vertices of two given grid cells. This operation could be useful in a variety of contexts such as linear extrapolation and interpolation, generating the code of a line whose slope is known, performing rational computations on the slopes of lines represented by their codes, and solving systems of linear equations.

The mode of operation here is quite different from that of the line recognizers described in V.2 and V.3. Initially, the positions of the two points to be connected by a line are found relative to each other. This is done by having one of the excited C's excite a chain of successive horizontal neighbors and the other a chain of successive vertical neighbors until the two chains meet. The numbers of C's in these vertical and horizontal chains are respectively the numerator and denominator of the slope of the line to be constructed. These integers

are stored in separate shift registers. These could be made out of the chains themselves, but this will not be done here both in order to simplify the logical design of the cells and to facilitate their use in later construction processes. Next, the code of the line is computed by synchronizing the two registers and counting the number of full cycles each makes. More precisely, suppose the slope is p/q. Then 1's in the code occur in columns $\lceil q/p \rceil$, $\lceil 2q/p \rceil$, ... q (see discussion of Theorem III.1). The kth 1 occurs in column $\lceil kq/p \rceil$, the number of complete cycles of the p register needed to equal or exceed k cycles of the q register. (A cycle of the p register consists of p single digit shifts.) Hence the code is obtained by simultaneously shifting both registers one digit at a time and writing a 0 each time the p register completes a cycle and a 1 each time the q register completes a cycle, omitting the 0 following in the latter case. Details follow.

There are three CS units, each connected to a different set of C's on the grid. CS_1 and CS_2 are to be used to set up and operate shift registers consisting of q and p excited C's on the grid. These CS units have identical state transition graphs as do the C's connected to them. However, there are no connections between CS_1 and CS_2. Furthermore, none of the C's connected to one of these CS's interacts with the other CS nor the C's connected to it. CS_3 is connected to C's that are part of the pattern to be constructed. These C's have no connections with CS_1 or CS_2 nor with any of the C's that connect to CS_1 or CS_2. CS^2 is connected to CS_1, CS_2, and CS_3. It synchronizes these CS's in determining the slope of and constructing the line.

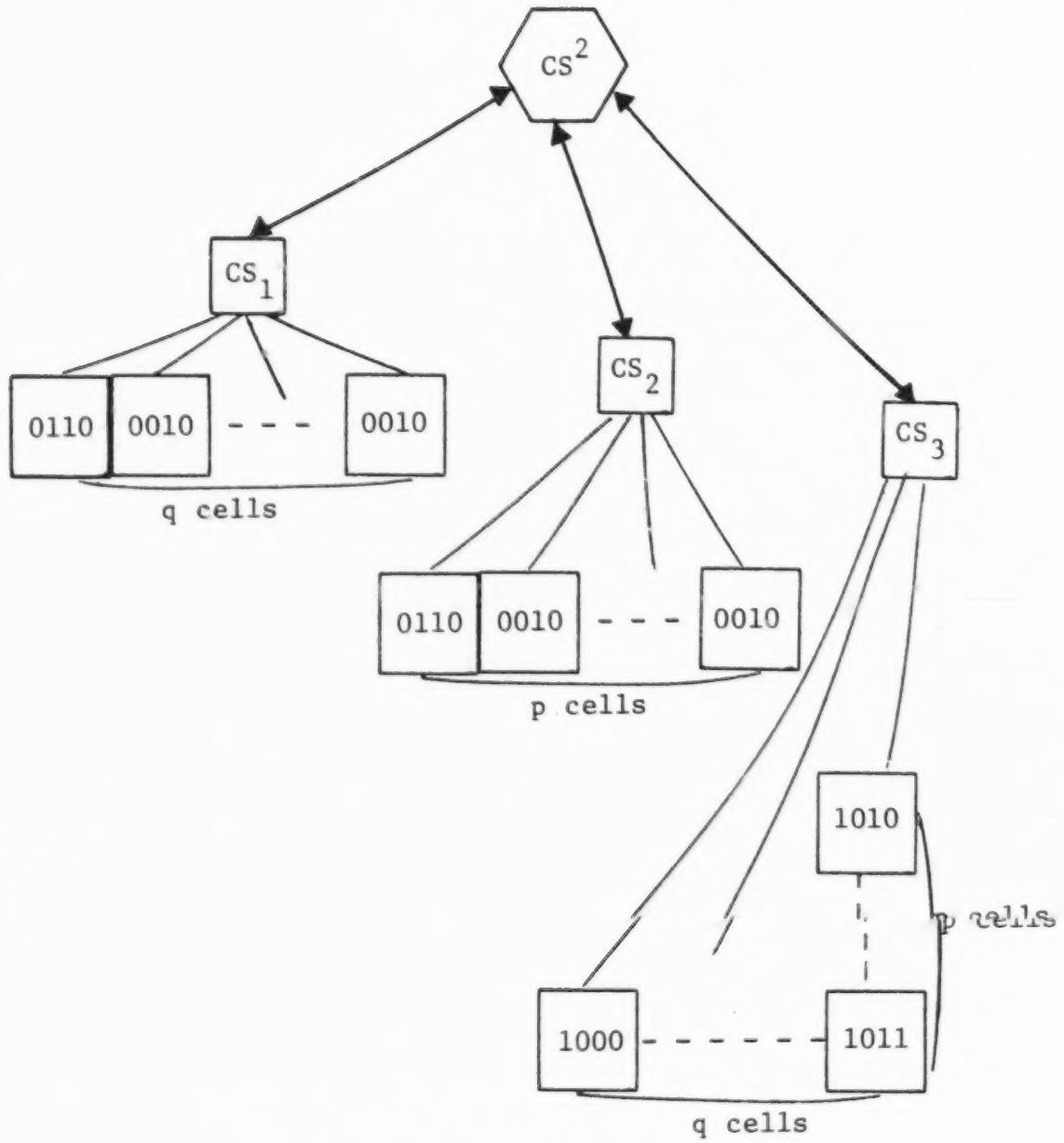


Figure V.5: Interconnections for Straight Line Generator

(See Figure V.5 for the interconnections between these units.)

Figure V.6 refers to the determination of p and q and the storing of their values in the shift registers. Initially CS_3 is in state 0 and two C's are connected to it. One of these is in state 1000 and is the grid square at which the line to be constructed starts. The other, q columns to the right and p rows above the C in state 1000, is in state 1010 and is the grid square at which the line is to end. CS_1 and CS_2 are both in state 0 and each is connected to a single C in state 0110. CS^2 is in state 0. From now on in this description consider only the case where $0 < p < q$. The other cases (changing the signs and reversing the inequalities) have the same transition graphs as the one discussed here, but different transition conditions and outputs derivable from those here by appropriate reflections of the specification of messages C's exchange with each other (discussed in V.2).

The operation starts with CS^2 sending 0 to CS_3, triggering the determination of p and q. CS_3 then sends 0 to the C's connected to it and 0 to CS^2. The C's respond by sending 0 to a single neighbor; the C in state 1000 to a right neighbor and the C in state 1010 to a lower neighbor. C's receiving these messages constitute the beginning of the chains of C's that will meet q units to the right of the C at which the constructed line is to start and p units below the C at which it is to end (see Figure V.5).

The chains are constructed as follows. A C receiving 0 from a left neighbor is in the horizontal chain and goes from state 0000 to 1001, connects to CS_3, and sends it 0 (denoted "con $\boxed{0}$ " in the transition

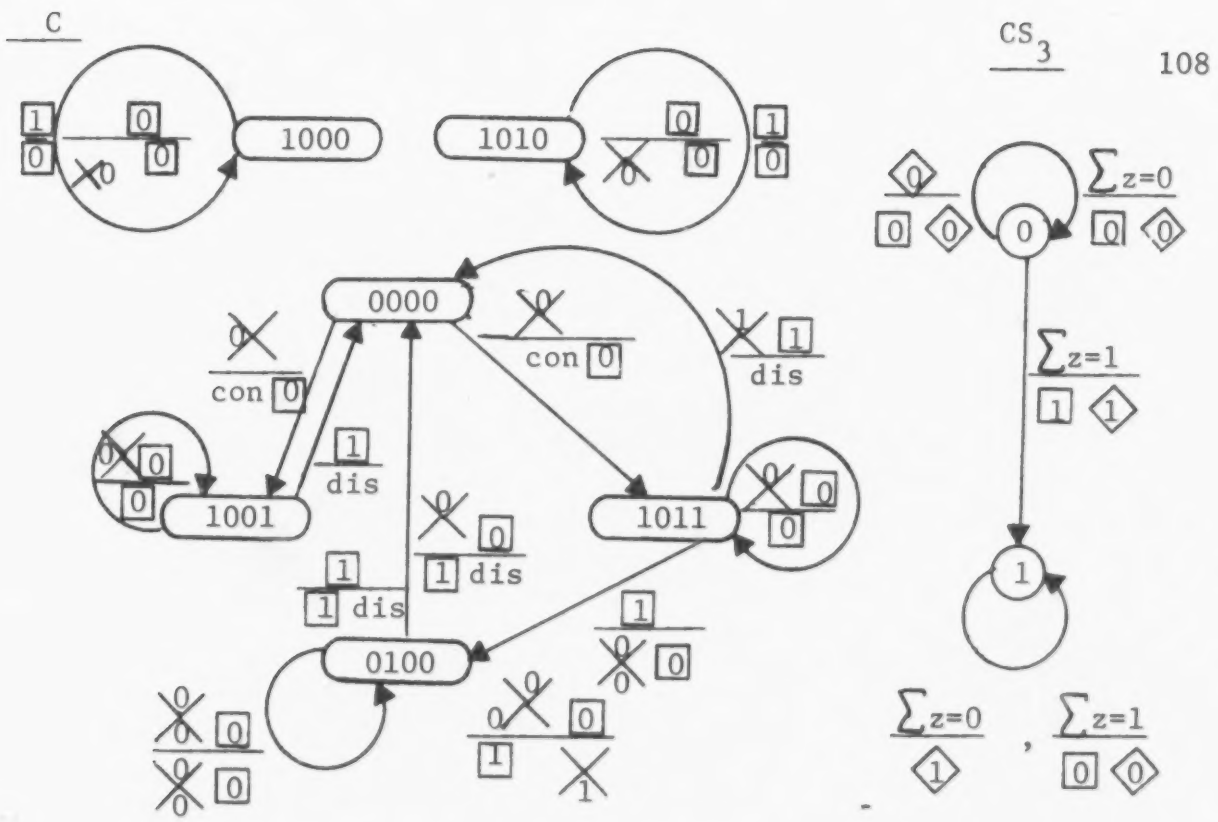


Figure V.6: Parallel Computer that Determines and Stores Slope of Line to be Constructed

graph). Similarly, a C receiving 0 from an upper neighbor goes from state 0000 to 1011 and sends 0 to CS_3. As long as CS_3 receives 0 from all C's connected to it, indicating that the two chains have not yet met, it again sends 0 to the C's, telling them to continue increasing the lengths of the chains. This increase is accomplished by the original C's (those between which the line is to be constructed) in states 1000 and 1010 sending 0 to their right and lower neighbor respectively. This 0 passes through the conducting C's that were added to the chains in the last step, to be received by C's in state 0000. The latter respond by going into the conducting state 1001 or 1011 on the end of the horizontal or vertical chain as described in the preceding sentences.

These steps are repeated as the chains spread from the original C's, intersecting after q steps. When this happens a C in the vertical chain in state 1011 receives a 0 from both an upper and a left neighbor, causing it to send 1 to its lower neighbor, 1 to CS_3 and go into state 0100. The 1 from this C to CS_3 causes the latter to go into state 1 and send 1 to CS^2 and to the C's connected to CS_3. It is the signal that q has been determined and p is about to be determined. First the 1 from CS_3 to its C's causes those in state 1001 to go into state 0000 and disconnect from CS_3, i.e., the C's in the horizontal chain are "erased" since they are no longer needed. At the same time C's in state 1011 below 0100 (i.e., receiving 1 from C's above) also go into state 0000 and disconnect from CS_3. The C in state 0100 sends a 1 to CS_3 and goes into state 0000 and the C's in state 1011 go into state

0100 and send 0 to their upper and lower neighbors and to CS_3.

The only C's now connected to CS_3 are the original two in states 1000 and 1010 and (p-1) C's in state 0100. The latter constitute the vertical chain whose C's must now be counted to determine the numerator of the slope. CS_3 in state 1 has just received 1's and 0's from the C's (indicating the horizontal and vertical chains spreading from the original C's have met) so it sends 0 to them and 0 to CS^2. Each C in state 0100 receiving a 0 from an upper and lower neighbor sends 0 to its upper and lower neighbors, 0 to CS_3 and stays in state 0100. C's in state 0100 receiving 0 only from an upper neighbor and CS_3 go into state 0000, send 1 to CS_3 and disconnect from it. CS_3, on receiving any 1's from its C's (indicating that there are still C's being erased and therefore the evaluation of p is continuing) sends them 0, sends 0 to CS^2 and stays in state 1. Thus, C's in state 0100 continue to "disappear" one at a time from the vertical chain. When the last of them has reverted to state 0000, CS_3 receives only 0's from its C's causing it to send 1 to CS^2 and remain in state 1. CS_1 and CS_2 were active throughout the activity just described in CS_3, receiving information via CS^2 in order to store the values of p and q. The following paragraph and Figure V.6 give the details of their operation.

In the beginning, CS^2 was in state 0 and sent 0 to CS_3, which responded by sending 0 to CS^2. CS^2 responded in turn by sending 0 to CS_1. CS_1 continued receiving 0's from CS^2 as long as CS_3 sent 0's to CS^2. CS_1, on receiving 0 from CS^2 sent 0 to the C connected to it which in turn sent 0 to its right neighbor. When a C in state 0000 received

0 from its left neighbor it connected to CS_1 and went into state 0010. On receiving 0 from CS_1, any C in state 0010 sent 0 to its right neighbor. Thus a sequence of q 0's from CS_3 to CS^2 resulted in a sequence of q 0's sent from CS^2 to CS_1 which in turn resulted in q C's in state 0010 connecting to CS_1. In this way, a chain of q C's was connected to CS_1, constituting a q digit register. Next, a single 1 from CS_3 to CS^2 caused the latter to go into state 1. This is the signal that the q register has been constructed and that the p register must now be constructed. Then, the p 0's from CS_3 to CS^2 caused the latter to say in state 1 and send p 0's to CS_2. The effect of these 0's from CS^2 to CS_2 was the same as the effect of the 0's from CS^2 to CS_1; namely, p C's in state 0010 became connected to CS_2 (see Figure V.5). At this point, the numerator and denominator of the slope of the line to be constructed have been determined and stored in the C's connected to CS_2 and CS_1. The next task is to compute the code from which the chain of excited cells will be constructed.

Refer now to Figure V.7, noting that states listed are the same as in Figure V.6. The graphs are separated only for clarity and are to be considered as superposed. On completion of the register construction described in the preceding paragraph, CS_3 sends 1 to CS^2. The latter responds by going into state 0 and sending 1 to CS_1 and CS_2. (Transition at bottom of Figure V.6.) These 1's to CS_1 and CS_2 are the signal to begin shifting the registers in order to compute the code. This is done by sending 00 to their C's, causing the C in state 0110 to send 1 to the corresponding CS and to its right neighbor (from now on, refer to Figure V.7). At the same time both CS_1 and CS_2 go into state 1 and

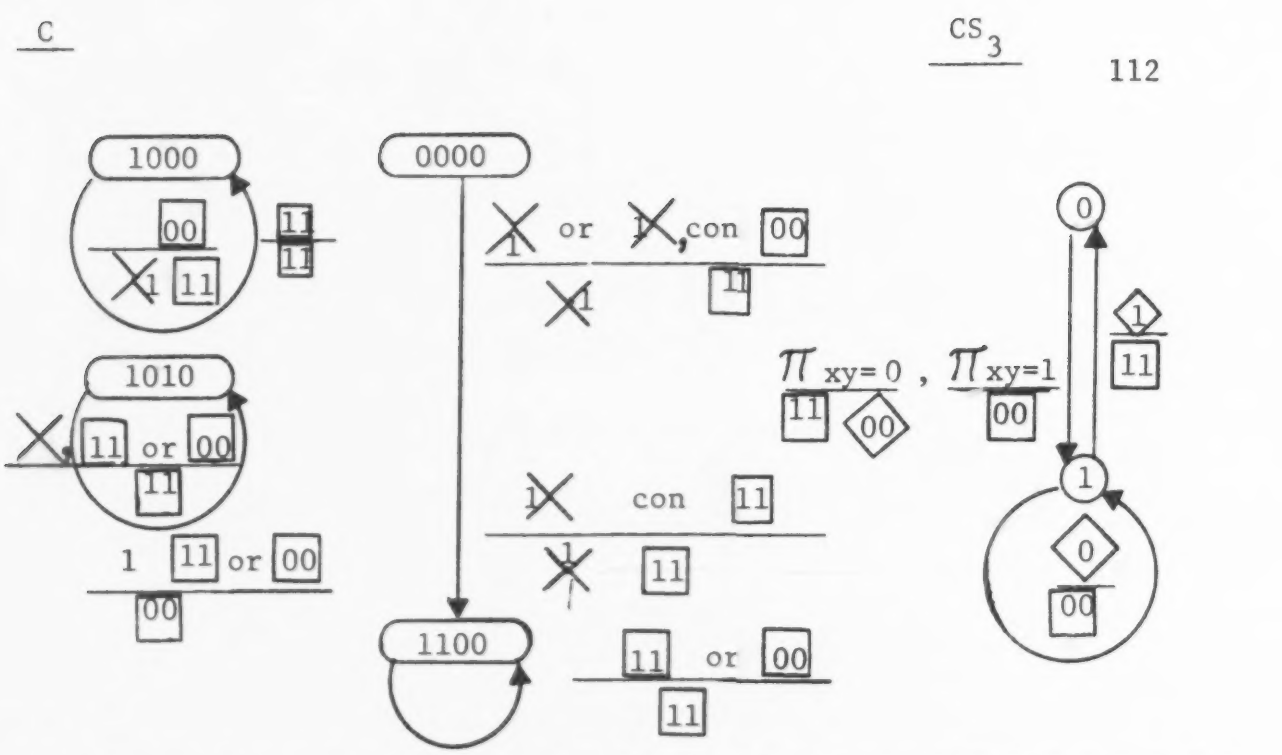


Figure V.7: Parallel Computer that Constructs a Line, Given its Slope

send 1 to CS^2. On receiving these 1's from CS_1 and CS_2, CS^2 stays in state 0 and sends 1 to CS_1 and CS_2. This exchange of 1's between CS_1 CS_2 and CS^2 corresponds to repeated shifting in the registers storing the values of p and q. This digit shifting in the C's connected to CS_1 and CS_2 is effected as follows. On receiving 1 from CS_1 (or CS_2) and nothing from a left neighbor, C's in state 0010 remain in that state and send 0 to their left neighbors and 0 to CS_1 (or CS_2). C's receiving 1 from a left neighbor and 0 from a right neighbor as well as 1 from CS_1 (or CS_2) send 1 to their right neighbors, 0 to their left neighbors and 1 to CS_1 (or CS_2). The 1 received from the left neighbor and sent to the right by such C's constitutes a single shift in the register. Since the C in state 0010 at the rightmost end of the registers does not receive 0 from the right (because it has no active right neighbors) its input is distinct from those that do. It is the only C that can receive a 1 from its left neighbor and nothing from its right neighbor. When this happens, the C in question sends 0 to CS_1 (or CS_2) signifying that a complete cycle of the register has just ended. For CS_1 the cycle is completed after q single digit shifts and for CS_2 after p single digit shifts as described above. At this time CS_1 (or CS_2) receives 0's from all the C's connected to it, causing it to send 00 to its C's and 0 to CS^2. The 00 to the C's starts the register shift over again. All succeeding exchanges between CS^2 and the two registers can be summarized as follows. When CS^2 receives any messages from both CS_1 and CS_2 it sends 1 to each of them. Thus, they continue shifting cyclically uninterrupted.

While the registers are shifting as described in the preceding paragraph CS^2 receives inputs from them which determine what it sends to CS_3 to construct the chain of C's to be excited. When both CS_1 and CS_2 send CS^2 1's, no message is sent to CS_3, because neither register is at the end of a shift and thus no code digit is forthcoming. Since $p < q$, the first change that occurs in messages from the registers is when a 1 is received from CS_1 (the q register) and a 0 from CS_2 (the p register). This corresponds to a 0 in the straight line code so a 0 is sent to CS_3. This condition will recur every p shifts of the p register (CS_2) until the q register completes a cycle (q single digit shifts), which corresponds to an occurence of a 1 of the straight line code. When that happens CS^2 receives a 0 from CS_1 and a 1 from CS_2 and goes into state 1. It remains in state 1 as long as it receives 1 from both CS_1 and CS_2. Finally it must receive 1 from CS_1 and 0 from CS_2; when it does it sends 1 to all three CS's and goes back into state 0, and the whole process continues. In this way CS^2 sends in sequence to CS_3 the digits of the code for a line of slope p/q.

Now consider the C's which are to constitute the constructed line. Before line construction starts CS_3 is in state 1 and is connected to only two C's; the C in state 1000 representing the beginning of the line to be constructed and the C in state 1010 representing the end of that line. On receiving the first 0 (corresponding to a code 0) from CS^2, CS_3 sends 00 to the two C's connected to it and remains in state 1. The C in state 1000 receives this 00 from CS_3 and sends a 1 to its right neighbor and 11 to CS_3. The C in state 1010 stays in the same state

until it receives 1 from a neighbor (indicating completion of the construction, to be described later). When CS_3 receives the next 0 from CS^2 it stays in state 1 and again sends 00 to the C's. Now a C in state 0000 receives the 1 sent by its left neighbor (in state 1000) in the preceding time step. This causes the C in state 0000 to go into state 1100, connect to CS_3, and send 1 to its right neighbor on receiving 00 from CS_3. In this way, a sequence of consecutive 0's from CS^2 to CS_3 results in the construction of a horizontal row of C's in state 1100.

When CS^2 sends 1 to CS_3, a bend (defined in II.2) is constructed as follows. First, CS_3 goes into state 0 and sends 11 to the C's. C's in state 0000 receiving 1 from a neighbor go into state 1100 and send 1 to an <u>upper</u> neighbor when they receive 11 from CS_3. Recall that such C's sent 1 to a <u>right</u> neighbor when they received 00 from CS_3. Thus, 11 from CS_3 to C's results in a vertical (upward) step in the construction and 00 results in a horizontal (rightward) step. In the next time step CS_3 goes back into state 1 and sends 00 to the C's, resulting in a horizontal step in the construction exactly as described in the preceding paragraph. In summary, 1's from CS^2 to CS_3 result in the construction of bends and 0's result in the construction of runs (defined in II.2). Eventually, the C in state 1010 receives 1 from a neighbor and sends 00 to CS_3, indicating that construction of the "line" connecting the two original C's is completed. CS_3 sends 00 to CS^2 which halts.

In conclusion, note that a trivial extension of the process that converts codes to "lines" of excited C's as described in the preceding paragraphs permits one to "draw" any curve whose code is known. The only additional information needed is the specification of run and bend types, which can be handled by providing appropriate "octant bits" (see II.2).

V.5 Discussion of Parallel Computers

A common mechanism in all the parallel computers discussed is the following. Messages are passed simultaneously through many sets of C's characterized by a particular set of states. When any of these messages reach a set of C's characterized by another set of states, this causes a CS to issue new instructions to the C's. This situation can be viewed as the simultaneous operation of a collection of generalized shift registers. In the straight line recognizer of V.2, code segments play the role of registers and 1's of the code are the source of the end-of-shift message. In the line recognizer of V.3 excited C's between those corresponding to digits recently transposed act as registers and C's corresponding to digits recently transposed are the source of the end-of-shift message. Finally, in the straight line generator of V.4 shift registers are used explicity to compute the code of a line. Although the use of registers is conventional in sequential computers, there are some novel features about their use in the parallel computers described here. Parts of the pattern being recognized act as registers, the number and size of registers is arbitrary, and both may be changed at any time by suitable instructions from CS's. Furthermore, many registers may be operated simultaneously under the control of a CS^2 to yield parallel computation. The kind of computation to be carried out is determined by the interconnections and transition graphs of C's, CS's, and CS^2's. For example, consider the following generalization of the straight line generator described in V.4 to a computer which multiplies (or divides) two rational numbers.

Let two rational numbers t/u and r/s be represented by the codes of lines with slopes t/u and r/s. If the 0's of these codes are represented by C's in a conducting state and the 1's by C's in the register state 0010 the result is two registers of length t and r, called R_t and R_r. If, on the other hand, all code digits are represented by register states the result is two registers of length u and s, called R_u and R_s. Now, let R_t and R_r be synchronized by an appropriate CS so that R_t shifts one digit each time a shift instruction is received from CS^2 but R_r shifts one digit only when R_t has shifted t digits. In this way the pair R_t, R_r goes through a complete cycle after $t \cdot r$ shift instructions from CS^2. Similarly, let R_u and R_s be synchronized so that the pair goes through a complete cycle after $u \cdot s$ shift instructions from CS^2. These two pairs of registers now play the role of the p and q registers of the straight line generator with the output now being the code for tr/us. In case tr > us the code of the reciprocal, us/tr, is generated by interchanging the roles of CS_1 and CS_2, the two registers of Figure V.7.

By slightly changing the ..ograms of the cellular automata described in the preceding sections it is possible to change their purpose from straight line recognition and generation to the detection of other geometric and topological properties of patterns of excited cells on the grid. In many cases it is only necessary to change some of the transitions without increasing the number of states. One of the simplest topological properties of patterns of interest in a variety of contexts is connectivity. In the next chapter is the design of a

parallel computer that identifies connected sets of excited C's on the grid and assigns each to a separate CS. Multiple connectedness, suitabl restricted to take grid resolving power into account, can also be determined.

CHAPTER VI: PARALLEL COMPUTERS TO RECOGNIZE CONNECTIVITY AND OTHER PROPERTIES

Two related properties of general importance in pattern recognition are connectivity and the presence of boundaries. The projection of a three dimensional object on a two dimensional surface is a connected region. Distinct objects project distinct (i.e., not connected to each other) images, unless they happen to lie along the same line of projection. The boundary of a connected region is important because it is the locus of transition between presence and absence of the image of an object. The parallel computers in this chapter detect the connectedness (VI.1) and find the boundaries (VI.2) of sets of excited C's on the grid. In addition, the design of a parallel computer that yields the approximation of boundaries by straight line segments is given in VI.3. The latter combines the line recognizer of V.2 with some new programs which could be useful in the recognition of polygons.

VI.1 Separation of Distinct Connected Regions

A connected region on the lattice will be defined here as a set of excited C's with the following property. There is a path from any C in the set through successive nearest neighbors in the set to any other C in the set. The parallel computer to be described here assigns each distinct connected region to a distinct CS. The mode of operation follows.

Initially, all excited C's are in a conducting state and connect to a single CS. Then, the CS instructs a single excited C connected to

it to send a message to all of its nearest neighbors. Since all excited C's are in a conducting state this message will spread to all C's in or bounding the connected region containing the source of the message and no others. By virtue of the conduction this spread occurs in a single step. After the message has spread throughout the connected region the excited (conducting) C's receiving it detach simultaneously from the original CS and attach to a new CS that henceforth interacts only with the C's of this single connected region. Meanwhile, the original CS continues to find connected regions in the set of remaining C's and assign them to distinct CS's until all C's have been detached from the original CS. Details follow.

Initially, all CS's are in state 0 and C's are in state 0000 and not tied to any CS's. There are no CS^2's in this design (see Figure VI.1). When the pattern is projected on the grid, C's excited by the pattern go into state 1001, attach to CS_0 and send 0 to it. CS_0, on receiving 0 from all C's attached to it, sends 01 to a single C and 1 to all others. The C receiving 01 responds by sending 1 to each of its four nearest neighbors, 1 to CS_0, and going into state 1000. C's in state 1001 receiving 1 from CS_0 send 0 back to it and stay in state 1001.

Now, CS_0 in state 0 receives a single 1 and some 0's from the C's and goes into state 1. The 1 it receives is an indication that message spreading has begun. CS_0 then sends 1 back to the C's. Since all excited C's are in the conducting state 1001, any C in the region connected to the C which started the message spreading (by sending 1 to its four nearest neighbors) receives 1 from some of its

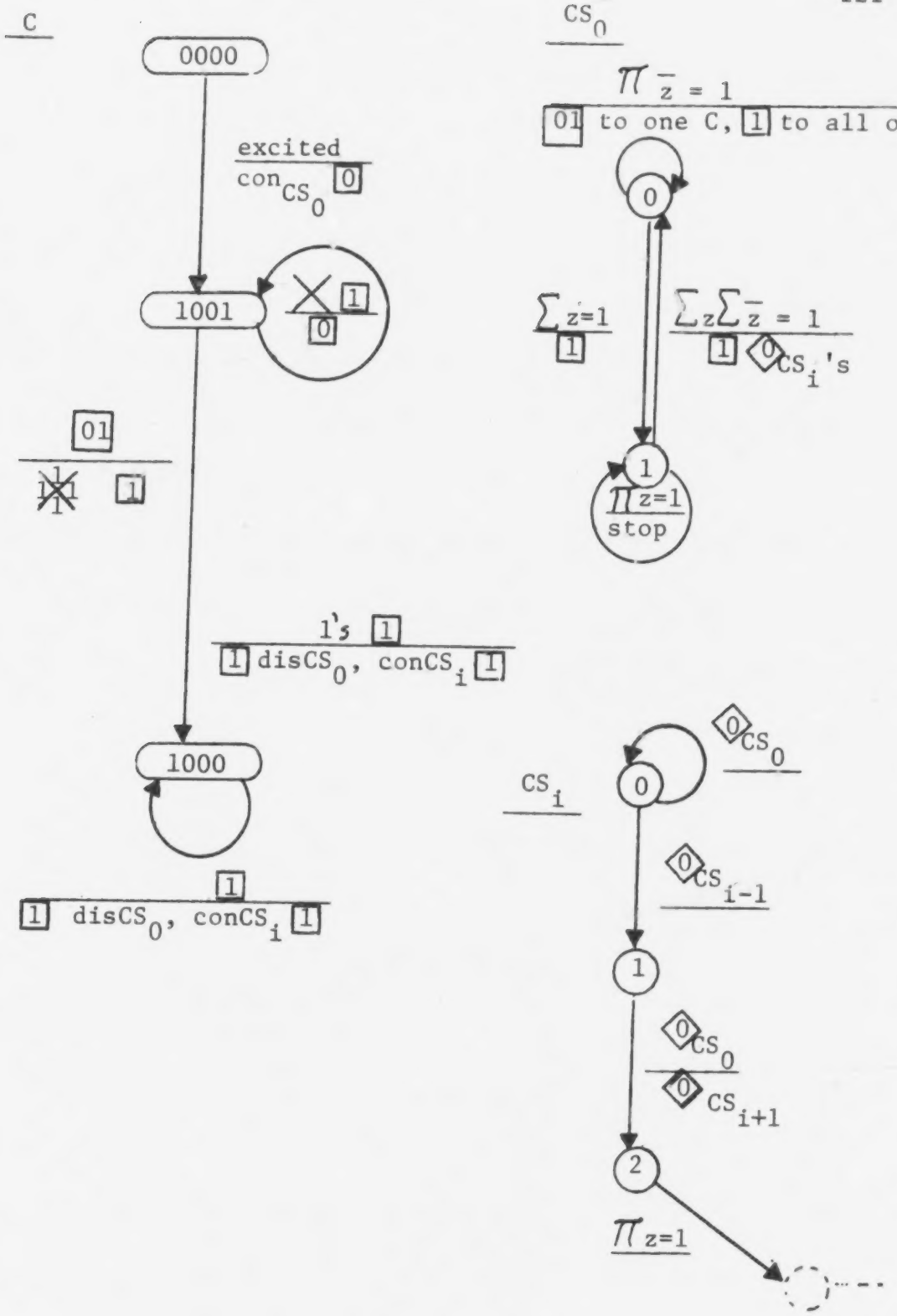


Figure VI.1: Parallel Computer that Recognizes Connected Regions

neighbors. Such C's respond to the simultaneous inputs of 1 from neighbors and 1 from CS_0 by sending 1 back to CS_0, and detaching from it. They then attach to CS_1, which is discussed below. C's in state 1001 outside the connected region receive no 1's from neighbors and thus remain in state 1001 and send 0 to CS_0.

The CS_i's constitute a collection of CS's with arbitrary (but fixed) capabilities, e.g., line recognizers. What is relevant here is the manner in which each connected region is attached to a distinct CS_i. Their transitions relevant to that operation are illustrated in Figure VI.1. The interconnections are shown in Figure VI.2. All CS_i's receive a common input from CS_C, and each CS_i sends an output to CS_{i+1}. There must be as many CS_i's as there are distinct connected regions on the grid.

Refer now to Figure VI.1. Initially CS_1 is in state 2, CS_2 is in state 1 and all other CS_i's are in state 0. The first input to the CS_i's occurs when C's detach from CS_0, i.e., when the first connected region has been found by CS_0. The C's in this region send 1's to CS_0 while C's in other regions attached to CS_0 send it 0. CS_0, in state 1, responds by going into state 0, sending 1 to the C's still attached to it and 0 to the CS_i's. Only CS_2, which is in state 1, is responsive to this input; thus it goes into state 2, ready to accept the C's from the next connected region to be detached from CS_0. On transition CS_2 sends 0 to CS_3, which causes the latter to go into state 1, priming it for later notification from CS_0 that a third connected region has been found. In this manner, assignment of each distinct connected region to

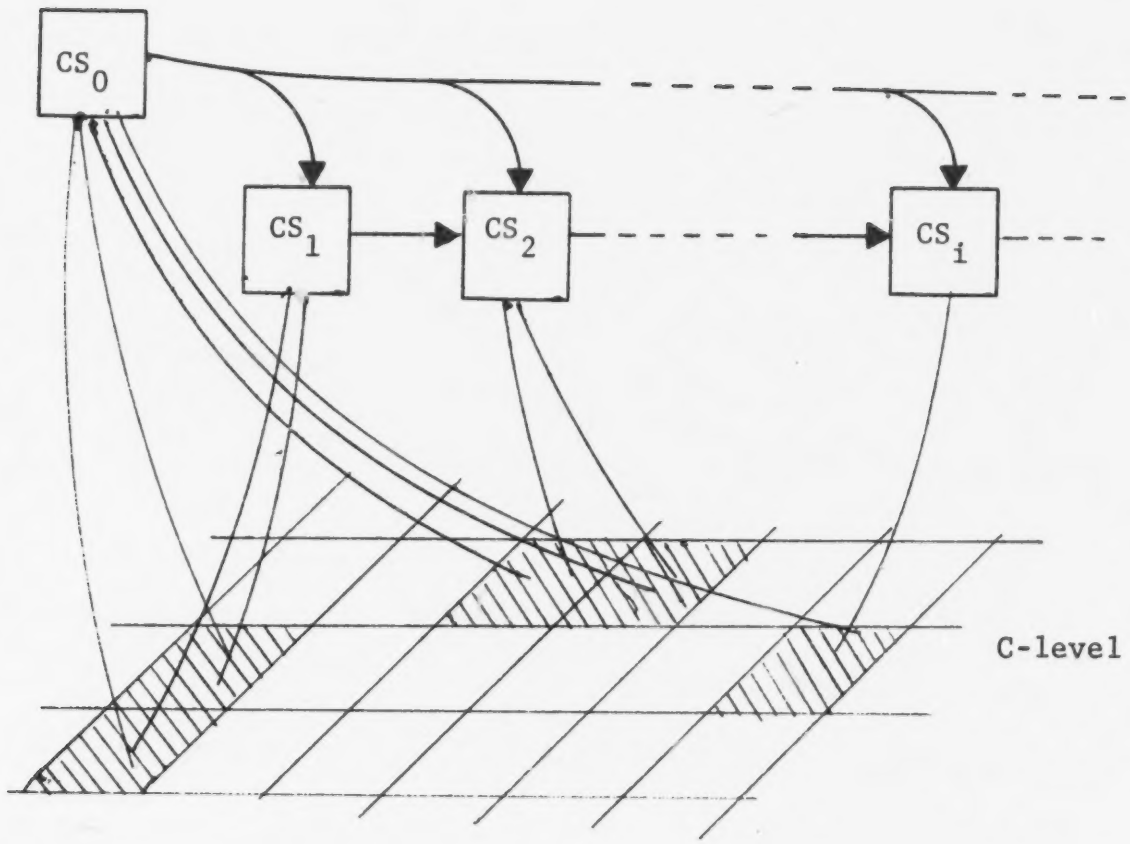


Figure VI.2: Interconnections for Assigning Regions to CS's

a distinct CS_i is propagated. CS_0 simply repeats the activities described until it receives only 1's from the C's attached to it, indicating that all C's have been accounted for as belonging to appropriate connected regions.

The multiplicity of connectedness (one plus the number of "holes" in a region) can be found by applying the program just described to the complement of a connected region on the lattice. The number of distinct connected regions (components) in the complement, i.e., the number of CS_i's to which C's are attached, is the multiplicity of connectedness. For a finite grid note that there is no essential loss in generality, from a topological point of view, if the region under consideration is taken as one containing none or a connected set of grid border cells. Doing this avoids bothersome discussion of multiple "pockets" between the region and the grid border. Note also that ambiguities with respect to topological characterizations of regions may stem from finite resolving power of the grid. They can be removed by using topologically equivalent regions. Often simple translation or magnification is sufficient.

The connectedness detector described in the preceding paragraphs has a much simpler design than the perceptrons, Turing machines, and iterative arrays designed to solve the same problem which are reviewed in part II of Minsky and Papert (1969). By virtue of the conducting C's, the operation here is more parallel as well. The programming and number of steps in computation are independent of grid size or the size and shape of the components. Only three time steps are required per component. During the first a message spreads in one connected region.

During the second C's in that region only respond to the spreading message. During the third they detach from CS_0.

VI.2 Determination of Boundaries of Connected Regions on the Grid

The boundary of a simple closed connected region in the plane contains a great deal of information about that region. For example, Green's theorem permits calculation of a surface integral in terms of a line integral over the boundary of the surface. If a computation can be carried out on the cells on the boundary of a connected region of excited cells on the grid instead of on each cell in that region, the substantial reduction in the number of cells involved can speed up the computation if much "sequentiality" is involved. Natural visual systems seem to be very sensitive to boundaries, perhaps in part for this reason.

A boundary can be given directly by specifying the cells it crosses (excites). In that case it can be considered part of either of two mutually exclusive sets which, together with the boundary, comprise the whole grid. It is generally convenient to regard a boundary so presented as part of the set with fewest grid border cells. In the following discussion the boundary is given by selecting the appropriate ones of a set of excited cells constituting a connected region. Note that this boundary is not the same as the boundary of its complement. The two boundaries are "neighbors", and thus coincide in the limit of vanishing cell size.

The following is the design of a parallel computer that effectively erases C's from the interior of a connected region of excited C's on the grid by sending them into a non-excited state. The resulting configuration is the grid's approximation to the curve bounding some simple closed connected region in the plane. This curve will be called the boundary in the following discussion and is not to be confused with the resulting configuration of excited C's.

At first sight, any C surrounded on all four sides by excited neighbors appears to be in an interior region and therefore subject to erasure. However, any C with exactly two excited nearest neighbors must have a boundary passing through two of its sides, i.e., those two sides it shares with neighbors. If one of these neighbors has four excited neighbors, one of its sides must be crossed by the boundary exiting its neighbor. Similarly, any C with only 1 excited neighbor must be crossed by the boundary through the side it shares with its neighbor. If this neighbor has four excited neighbors, it must nevertheless be crossed by the boundary through the side it shares with its neighbor. Thus, to erase only C's that do not contain the boundary, those C's with four nearest neighbors adjacent to C's with one or two nearest neighbors must not be erased. Figure VI.3a illustrates this situation.

Figure VI.3b is the transition graph of the parallel computer that erases all four-neighbor C's that are not adjacent to one- or two-neighbor C's. (A C with exactly j excited nearest neighbors is called a j-neighbor C here.) Initially all C's are in state 0000 and CS is in state 0. When the pattern is projected on the grid, all C's excited

a)

b)

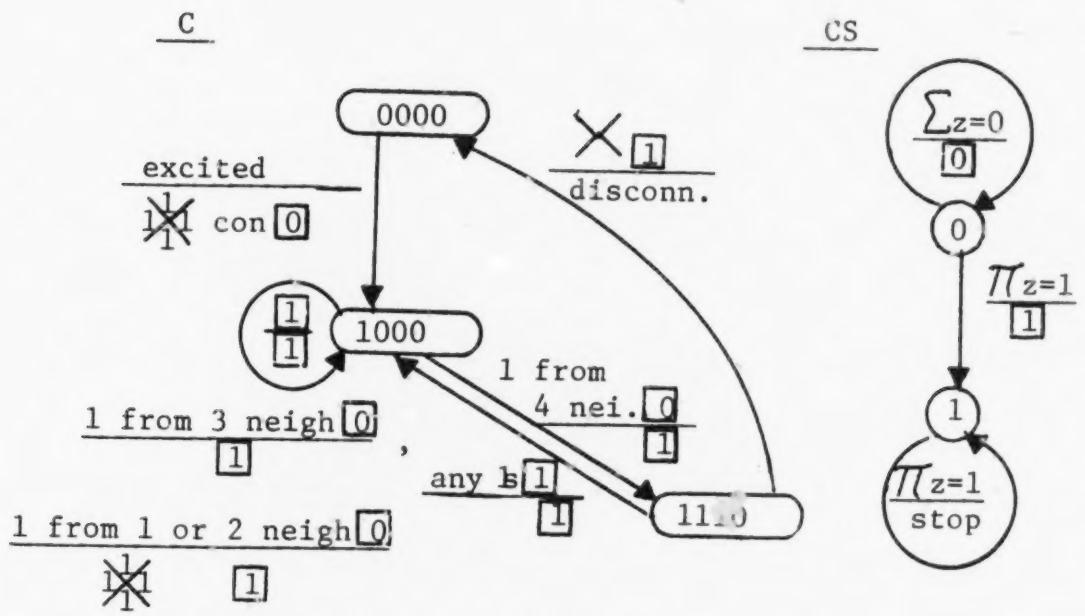


Figure VI.3: Parallel Computer to Recognize "Boundaries"

by it go into state 1000, send 1 to all four nearest neighbors and 0 to CS. CS then stays in state 0 and sends 0 to the C's connected to it. Those C's receiving a 1 from three neighbors and 0 from CS stay in state 1000 and send 1 to CS. Simultaneously, C's receiving a 1 from one or two neighbors stay in state 1000, send 1 to CS and 1 to all four of their neighbors. Those C's receiving a 1 from all four neighbors and 0 from CS go into state 1110 and send 1 to CS.

When CS receives 1's from the C's it goes into state 1 and sends 1 back to the C's. C's in state 1110, (with four excited nearest neighbors) respond to this 1 from CS by going into state 1000 if they receive a 1 from any neighbors and into state 0000 and disconnecting from CS otherwise. The interior C's are thereby "erased". All other C's simply stay in state 1000 when they receive the 1 from CS and send 1 back to CS. CS then halts, and the process is complete.

VI.3 Straight Line Approximation of Curves

The following is a generalization of the straight line recognizer described in V.2. Since there will be several CS's operating together, the line recognizing CS is called CS_L. The purpose of this design is to partition a chain of excited C's that does not represent a straight line into a sequence of chains, each of which corresponds to a straight line segment. The result of computation is the grid's approximation by straight line segments of a curve projected on it.

CS_L differs slightly from the recognizer in V.2. If a violation of the segment length constraint of Theorem II.2 occurs, CS_L does not simply stop and reject the pattern as a non-straight; it halts to

disconnect just enough C's so that those remaining do not violate that constraint. It then resumes CCF reduction on the remaining string. Details follow.

We consider the special case where the input consists of a chain of 1 and 2-neighbor C's whose run and bend configurations are compatible with a single octant. General inputs are discussed in VI.4.

There are three CS's and one CS^2 in this design. CS_L performs the CCF reduction, CS_{str} determines the substring of C's to be detached from CS_L when a violation of the segment length constraint occurs, and CS_h holds those C's which are detached from CS_L by CS_{str}. CS^2 is used to start and stop the entire process (see Figure VI.4 for connections between these units). Since the nearest neighbor configurations of the C's are compatible with a single octant, the transition graphs in the following discussion are for octant I. The graphs for other octants may be derived by reflecting the specification of nearest neighbor messages between C's exactly as described in V.2.

Refer now to Figure VI.5. Initially all C's are in state 1000 and connected to CS_h which is in state 0. CS_{str} and CS^2 are in state 0 and CS_L is in state 2. CS^2 starts the computation by sending 1 to CS_h which then sends 01 to all the C's connected to it. On its receipt the C's send 1 to each of their four nearest neighbor and 0 back to CS_h. CS_h then sends 1 to CS_L and 0 to the C's, which makes them determine their nearest neighbor configurations on the basis of the 1's sent them in the previous time step. Just as in the straight line recognizer described in V.2, run C's go into state 1000, bend C's into state 1100

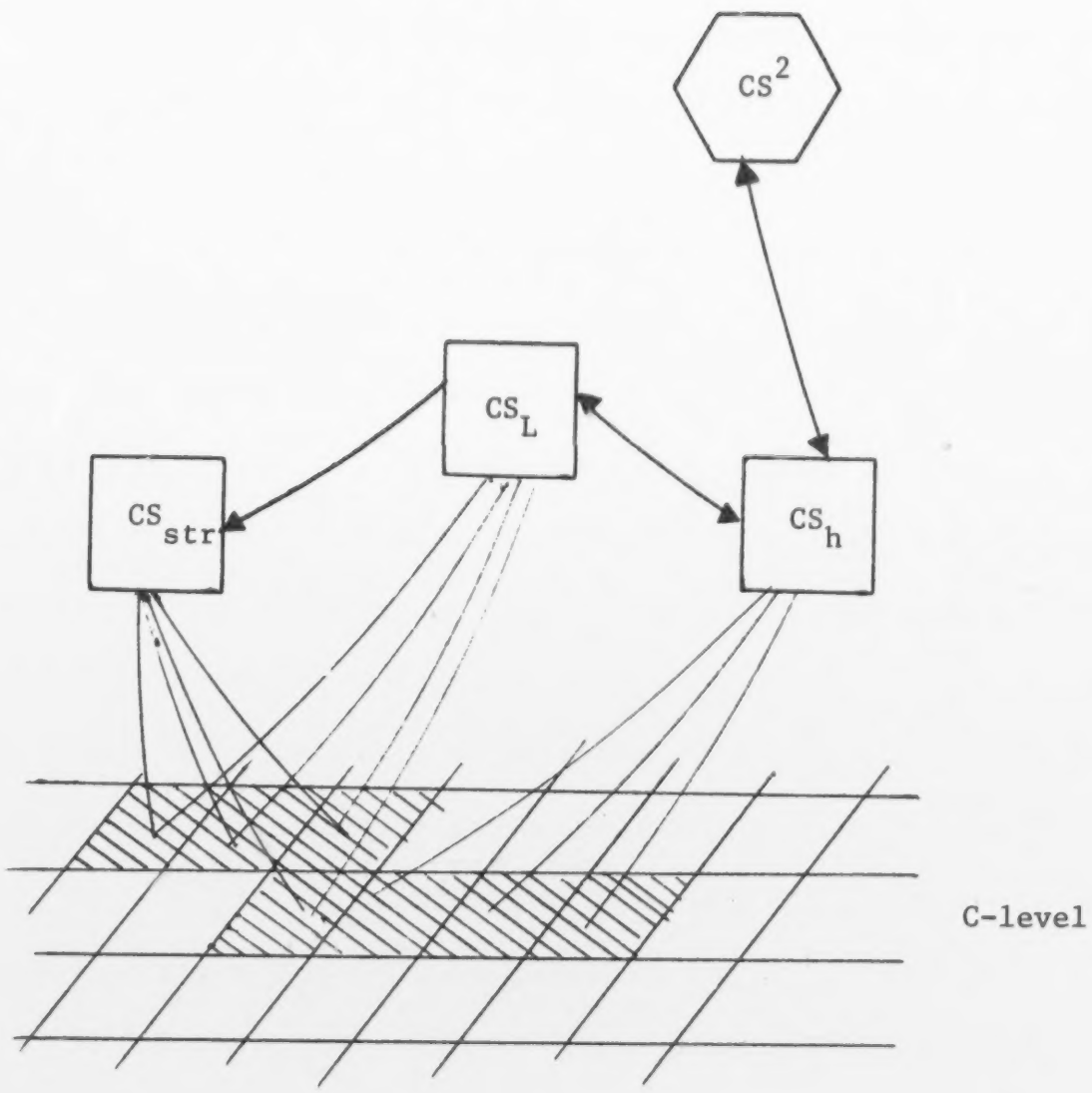


Figure VI.4: Interconnections for Straight Line Approximation

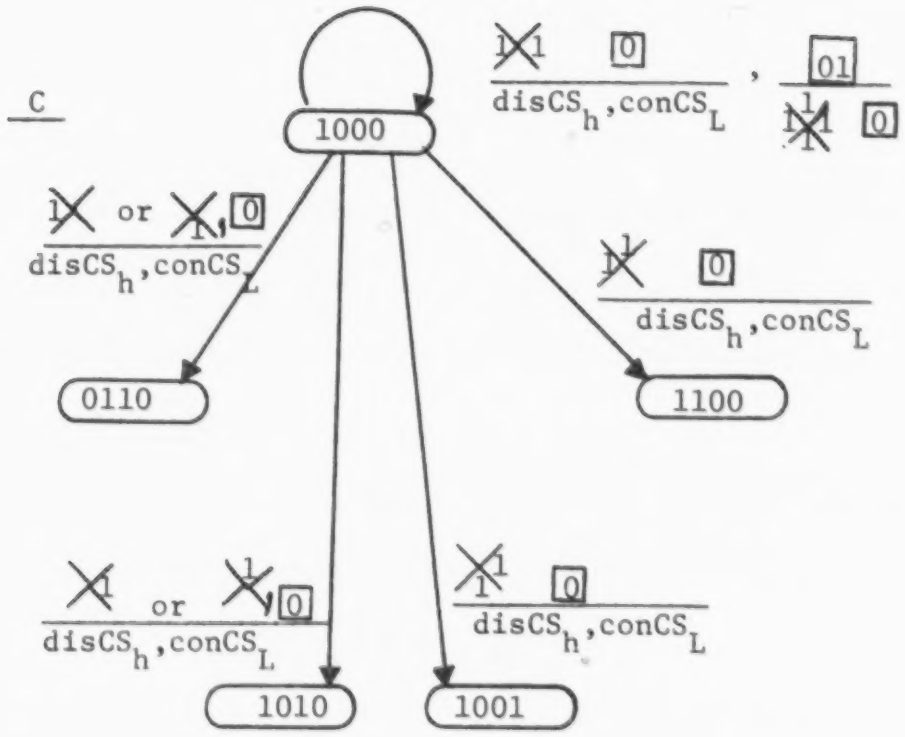


Figure VI.5: Parallel Computer that Holds C's not Currently Being Processed by Line Recognizer or Illegal String Truncator

and conducting C's into state 1001 in response to the 0 from CS_h. That is, the second digit of the states of run and bend C's corresponds to the appropriate straight line code digit. The C's terminating the chain, i.e., those receiving a 1 from only one nearest neighbor, are handled differently in order to distinguish which end of the chain of C's each lies on. This is done by sending the right end C (the C receiving 1 from a left or lower neighbor) into state 0110 and the left end C (receiving 1 from an upper or right neighbor) into state 1010. As soon as the C's go into their run, bend, or end states they detach from CS_h and connect to CS_L ("$disCS_h$" and "$conCS_L$" in Figure VI.5). CS_L is in state 2 and receiving 1 from CS_h. It stays in state 2 and sends 1 to the C's. The C's respond as they did in the line recognizer of Figure V.3, i.e., a 0 is deleted from each code segment. The further behavior of CS_L is the same as that of CS in Figure V.3, i.e., 0's are deleted from segments and 1's terminating long and short segments are rewritten as 0's and 1's respectively when there are no more 0's to be deleted. Refer now to Figure VI.6. Except for the transitions shown there, those of CS_L are the same as those in Figure V.3. This avoids repetitious discussion of Figure V.3 and shows what changes are needed to yield the present design.

There are two major changes. First, the messages 0 and 11 cause the C's to detach from CS_L and connect to CS_h or CS_{str} respectively. Second, this also applies to C's in the conducting state, 1001. They do not disconnect from CS_L on digit deletion. We now discuss the CS design modifications necessitated by these changes.

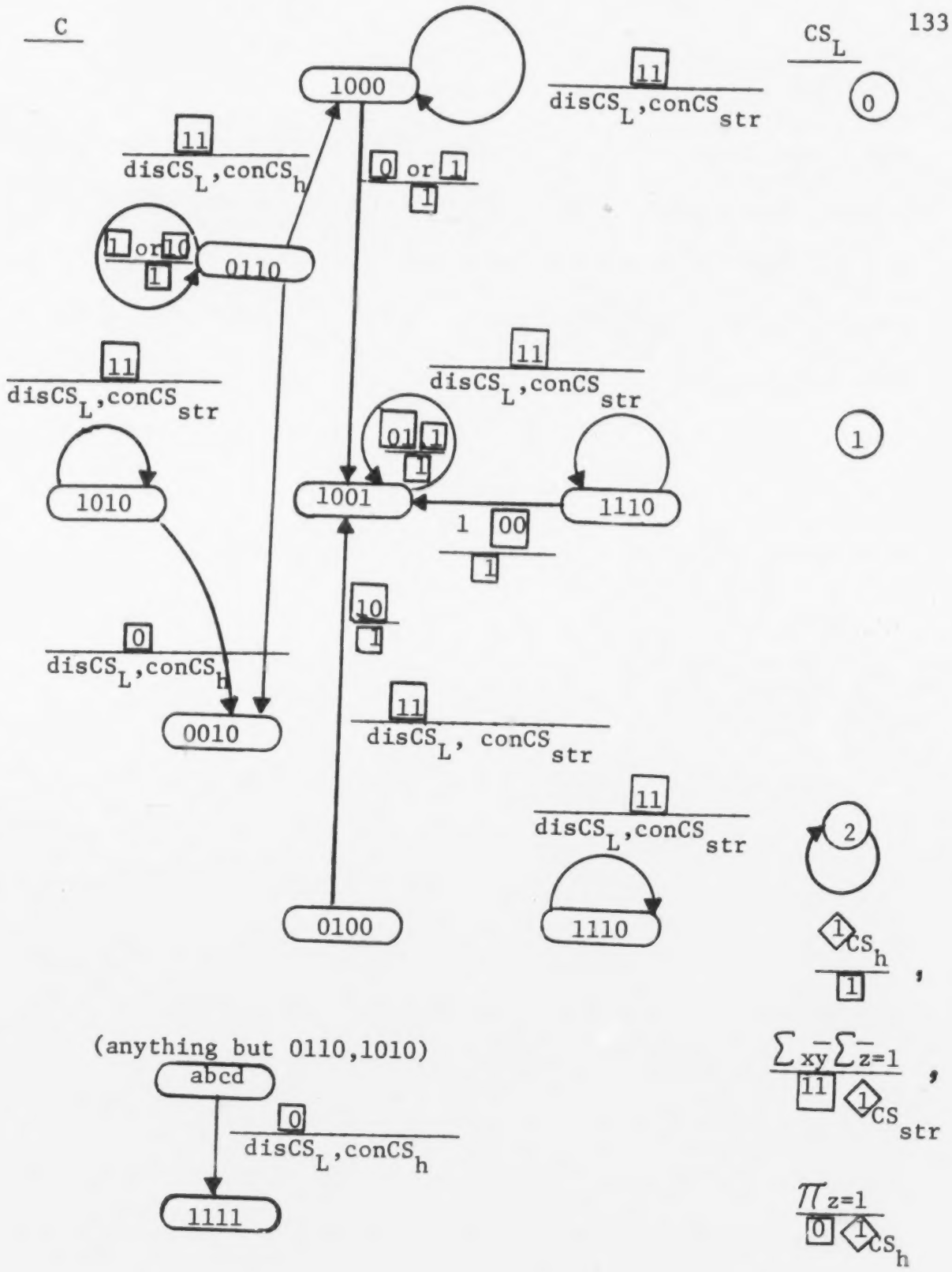


Figure VI.6: Straight Line Recognizer Used in the Approximation of Curves by Line Segments

The first new mode of operation occurs when CS_L receives a 10 and 0 simultaneously from its C's, indicating there is a violation of the segment length constraint of Theorem II.2. Instead of halting and rejecting, CS_L to sends 11 to the C's connected to it and 1 to CS_{str}. The C's disconnect from CS_L and connect to CS_{str}, which then determines where the string can be truncated in order to eliminate violation of the segment length constraint. The C in state 0110 is an exception. It is on the right end of the chain and so must be in the substring rejected by truncation. It goes into state 1000, disconnects from CS_L and connects to CS_h.

We now describe what happens after the C's connect to CS_{str}. A three letter alphabet is introduced representing their states. Strings over that alphabet, representing a left-to-right ordering of the excited C's, are then processed to truncate the "illegal" part.

Let 0 stand for C's in state 1000 (code 0's), 1 for C's in state 1100 (code 1's) and 2 for C's in state 1110. A string is generated by successively tracing the C's connected to CS_{str} starting at the left end C (in state 1010) and proceeding through nearest neighbor C's to the right end C (in state 0110). The empty symbol is written for a C in the conducting state (1001) and the end states (1010 and 0110). Intuitively, these symbols may be regarded as a length classification of code segments, i.e., 0 and 1 represent long and short segments respectively and 2 represents a "too short" segment. Strings not violating segment length constraints of straight line codes (called legal strings here) consist of 0's and 1's or 1's and 2's or strings of a single digit. The set of

legal strings is given by the regular* expression (0 + 1)* + (1 + 2)*. Strings violating the segment length constraint (called illegal strings) contain both 0's and 2's. Since an illegal string necessarily contains both 0's and 2's, the first occurrence of the second to appear in the string marks the beginning of the portion to be truncated. The legal portion is reconnected to CS_L (detached from CS_{str}) and the illegal portion to CS_h.

Refer now to Figure VI.7 which shows how CS_{str} accomplishes the foregoing. CS_{str} (in state 0) receives 1 from CS_L, stays in state 0, and sends 0 to the C's connected to it. The left terminating C (in state 1010) sends 0 to its right and upper neighbors, C's in state 1100 (string symbol 1) go into the conducting state 1101, and all other C's remain in their current states. All C's send 0 (not to be confused with string symbol 0!) to CS_{str}, which then sends 1 to all C's and goes into state 1. Suppose 0 is the first of the symbols 0 and 2 to occur in the string (the other case will be described later). Then the leftmost C in state 1000 (leftmost string symbol 0) will receive the message 0 sent by the left terminating C (in state 1010) in the preceding time step. This 0 from a neighbor and 1 from CS_{str} causes the C to send 1 to CS_{str} and remain in state 1000. All other C's remain in their current states and send 0 to CS_{str}.

When CS_{str} receives 0's and 1 from C's it goes into state 2, and sends them 10. C's in state 1000 (string symbol 0) then go into the conducting state 1001 and all others stay in their current states and send 0 to CS_{str}. At the same time, the left terminating C (in state 1010)

---

* See Chapter 4 of Ginzburg (1968) for definitions and notation of regular expressions.

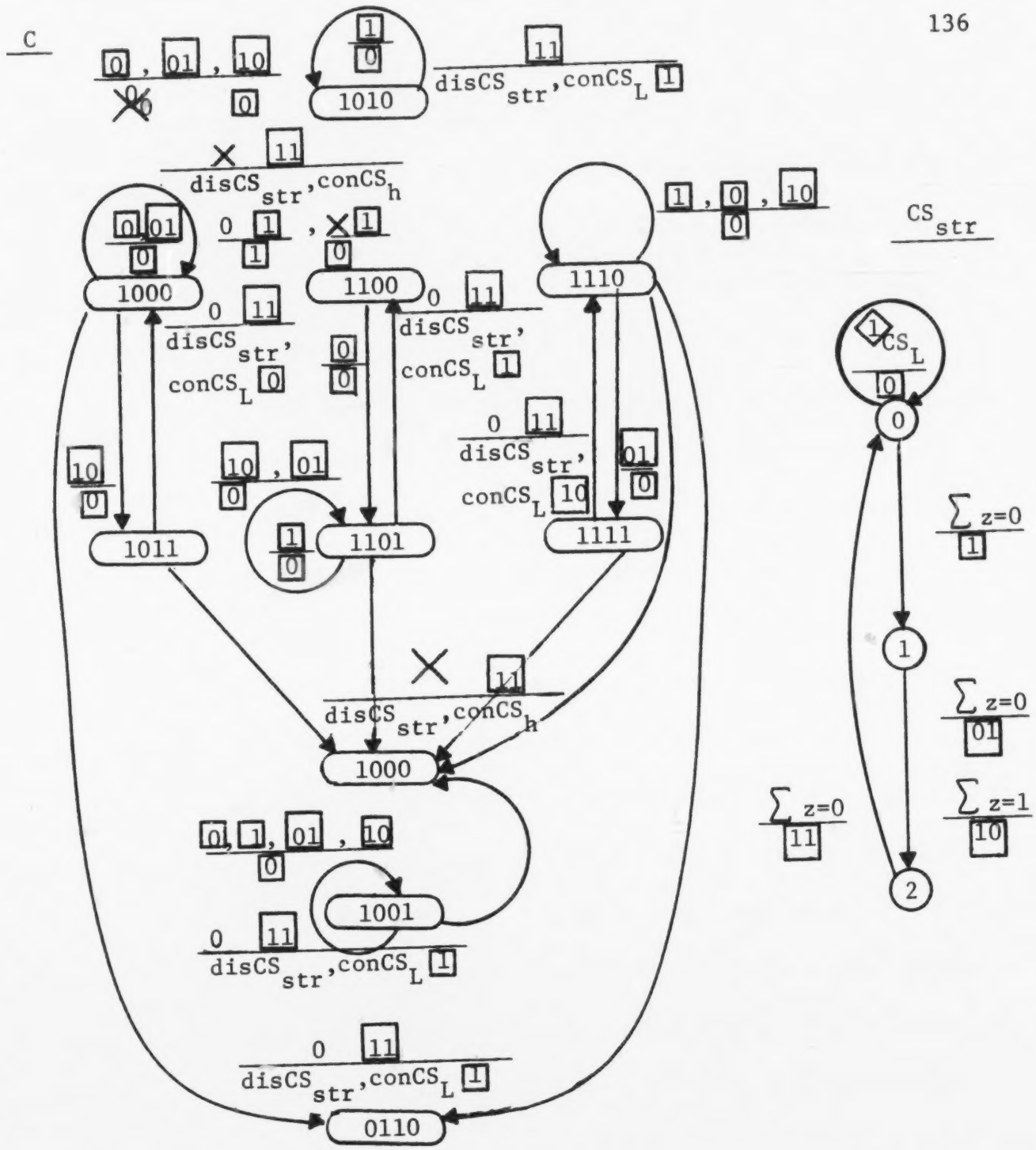


Figure VI.7: Parallel Computer that Truncates Illegal Substring

sends 0 to its right and upper neighbors. Since all C's corresponding to string symbols 1 and 0 are now in conducting states, the 0 sent from the left terminating C in the preceding sentence will be conducted through the chain of C's to the leftmost C in state 1110 (string symbol 2). The longest legal substring starting from the left then corresponds to the set of C's receiving this conducted 0. The transition graph (Figure VI.7) expresses the preceding as follows. CS_{str}, in state 2 and receiving 0 from all C's, sends them 11 and goes back into state 0. The left terminating C and all C's that receive 0 from it and 11 from CS_{str} disconnect from CS_{str} and connect to CS_L. Those in conducting states revert to their original nonconducting states (i.e., the rightmost digit of their states changes from 1 to 0). On connecting to CS_L, the C's send it the messages they would send if CCF-reduction were in progress. From the discussion of Figure V.3 recall that during CCF-reduction C's in state 1000 send 0, those in state 1100 or 1010 send 1. In addition, the C in state 1110 which is connecting to CS_L goes into state 0110 and sends 1 to CS_L. CS_L, in state 2, resumes CCF-reduction as described in V.2.

C's corresponding to symbols in the truncated (illegal) portion of the string did not receive 0 from a neighbor when CS_{str} sent them 11. Thereupon they go into state 1000, detach from CS_{str} and connect to CS_h at the same time as the C's of the legal substring connect to CS_L.

Now consider the case where 2 is the first of the symbols 0 and 2 to occur in the string. Consider Figure VI.7 when CS_{str} is in state 1 and C's corresponding to code symbol 1 are conducting. When the left terminating C (in state 1010) sends 0 to its upper and right neighbors,

no C in state 1000 (string symbol 0) will receive it. Thus, CS_{str} receives only 0's from the C's instead of the 0's and 1 it received when 0 was the first of the symbols 0 and 2 to occur in the string. In this case, CS_{str} sends 01 to the C's when it goes into state 2. C's in state 1110 (string symbol 2) go into the conducting state 1111. Except for this difference (namely C's corresponding to 2's conduct instead of C's corresponding to 0's), truncation of the illegal string proceeds exactly as described in the preceding case. Notice that this truncation process requires only four time steps no matter how long the string is, corresponding to the successive transitions of CS_{str} into states 0, 1, 2, and 0.

The truncation process just described is repeated every time CS_L receives 10 and 0 simultaneously from the C's connected to it, i.e., when the segment length constraint of Theorem II.2 is violated. Eventually, the set of C's connected to CS_L constitutes a chain compatible with a straight line. This occurs when CS_L, in state 2 receives only 1's from its C's, indicating CCF reduction is complete. Recall that in the line recognizer described in V.2 the completion of reduction caused CS to halt. Here, however, recognition of the substring connected to CS_L may be complete but the rejected substrings, attached to CS_h, need processing. The C's disconnect from CS_L and connect to CS_h. Those in states 0110 and 1010 (C's on the right and left ends of the chain recognized as a straight line by CS_L) go into state 0010. They correspond to the ends of the straight line segment represented by the intervening C's. The latter go into conducting state 1111.

The message 1 sent from CS_L to CS_h acts as a request for new C's to be processed. It causes CS_h (refer to Figure VI.5) to send 01 to its C's. Those in state 1000 then send 1 to all four nearest neighbors and 0 back to CS_h. This causes CS_h to send 0 to the C's which then determine their nearest neighbor configurations and connect to CS_L, and repeat the process described in the opening paragraphs of VI.3. The C's in state 0010 and 1111 connected to CS_h have already been processed by CS_L; therefore they do not disconnect from CS_h but simply send it 0 when they receive 0 from it. When the only C's attached to CS_h are those that have been recognized as belonging to a straight line segment by CS_L, CS_h receives only 1's from the C's and the entire process is complete. CS_h then sends 1 to CS^2 which goes into state 1 and halts, signifying that the given chain has been approximated by straight line segments. The C's in state 0010 connected to CS_h correspond to vertices in this polygonal approximation of the chain, and a set of adjacent C's in state 1111 corresponds to a straight line segment approximating part of the original chain.

The interplay of the CS's described in the preceding paragraphs may be summarized as follows. CS_L carries out straight line code reduction on the collection of C's connected to it. When an error condition arises, indicating that the C's cannot have been excited by a single straight line, all C's disconnect from CS_L and connect to CS_{str}. The latter then finds the longest substring of C's, starting from the left, that does not violate a straight line constraint (a necessary but not sufficient condition for a straight line). This legal string of C's is returned to the control of CS_L which resumes code reduction; and the

C's truncated from the string are connected to CS_h where they await further instructions. These instructions come when CS_L has successfully completed code reduction on all C's connected to it. These latter C's (representing the substring of reduced code) are disconnected from CS_L and connected to CS_h. Next, as yet unprocessed C's connected to CS_h are detached from it and connected to CS_L whereupon code reduction commences on this new group of C's. When the entire chain has been reduced to straight line subchains in the above manner, the process is finished.

VI.4 Aspects of Polygon Recognition

Several of the parallel computers described in Chapters V and VI could be combined to recognize more complex patterns, such as those consisting of collections of lines. Polygon recognition is an example of such a task. Two kinds of complexity introduced by considering such patterns are the increase in number of parts to be processed and the interaction of those parts in regions where the C's excited by them overlap. The increase in number of parts of the pattern will be discussed in the following paragraph and the interaction of parts later.

If a pattern consists of a number of disconnected parts, they can be assigned to separate CS's by the connected regions computer described in VI.1. Suppose each of these parts is a line to be recognized by a CS such as described in V.2. The computation can be carried out either by simultaneously applying as many CS's as there are lines or by applying one CS to the lines sequentially. The former approach can use an array of CS's associated with the layer of C's. The structure of the CS's is

even simpler than the C's (fewer states and transitions). Since each line projected on the grid excites many C's, the number of CS's necessary to simultaneously recognize many lines is correspondingly less than the number of excited C's. The time required for recognition is the maximum required for any of the individual lines. If the same CS is used for each line in succession, the total recognition time is the sum of the time required for each line. As each line recognition is highly parallel (see V.2), computation time generally remains much less than for a sequential scan of C's even in this case.

Consider now the projection on the grid of two polygon sides that intersect at a vertex (Figure VI.8). If the angle these sides make at the vertex is 180^{o}, the single line they constitute can be recognized by the simple CS of V.2. If the angle is less than 180^{o}, but still so obtuse that both lines have slopes compatible with a single octant, the more complex combination of CS's of VI.3, is needed to recognize the pair (Figure VI.8b). Making the vertex angle more acute yields slopes of the adjoining sides that must be in two distinct octants (Figure VI.8c). Recognizing this case requires a CS that operates much like CS_{str} of VI.3. That is, a legal (compatible with a single octant) string here is a string of runs of one type and bends of one type. Such strings can be found by sending a message that conducts along the chain from a C on one end of the chain. Just as the CS_{str}, the chain is truncated at the C where the conducted message first encounters an illegal symbol, i.e., a run or bend type conflicting with those already encountered. The substring to the left of the illegal symbol consists of runs and bends compatible with a single octant and can

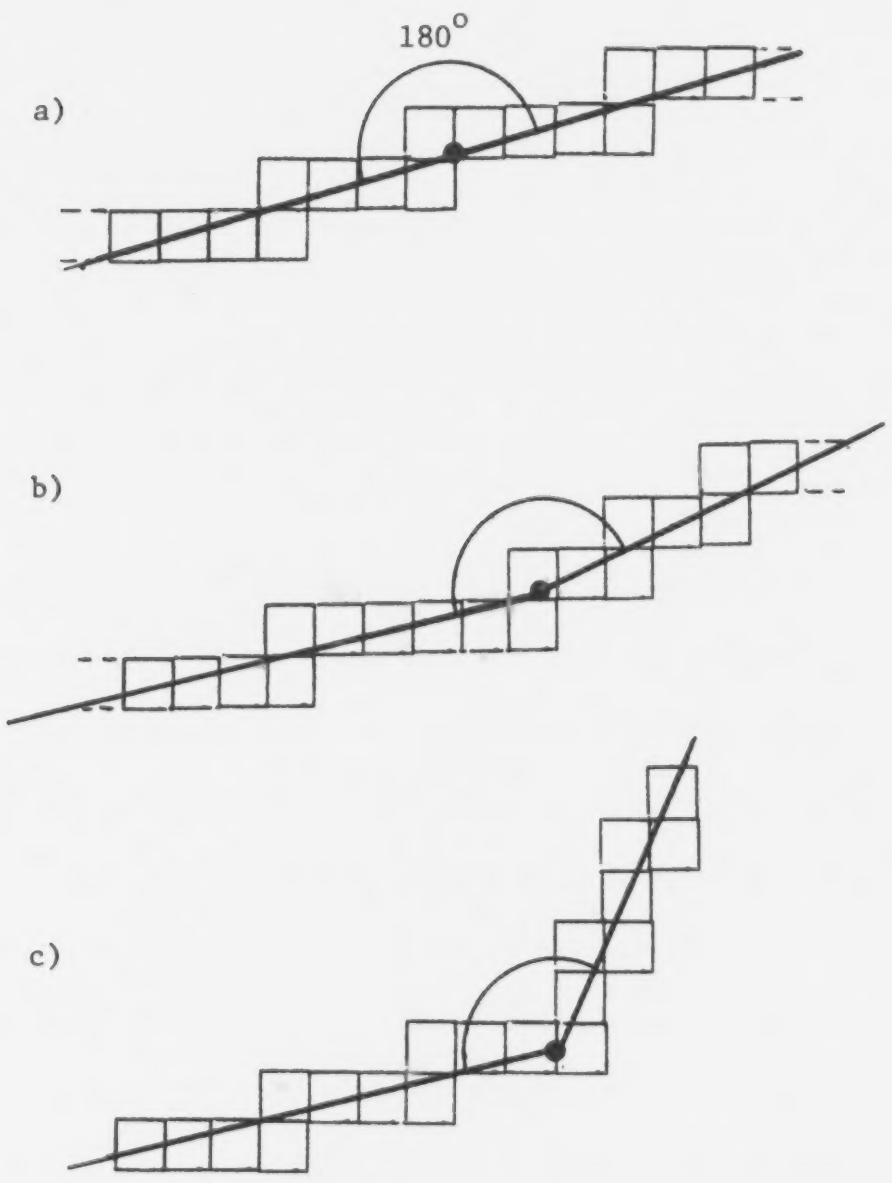


Figure VI.8: Grid Cells Crossed by Lines That Meet at Obtuse Angles

be connected to a line recognizer like CS_L in V.3. The string to the right of and including the illegal symbol is then processed exactly as the original string. The details and transition graph are omitted as they are similar to those of CS_{str}. Now suppose the vertex angle is so acute that the nearest neighbors of the C's excited by one of the sides are excited by the other side as well. Figure VI.9 illustrates such a case for a line whose slope is in octant I; the C's excited by the side in question are boldly outlined. The C's near the vertices, excited by the adjacent sides, are more faintly outlined. The close approach of the lines to each other near the vertex converts some 2-neighbor C's to 3- and 4-neighbor C's.

If the path of the lines through excited C's is not known a priori but is to be inferred from the configuration of excited C's, there is an ambiguity as to which of these 3- and 4-neighbor C's it traverses. That is, a path traced through a 2-neighbor C must enter through the side shared with one excited neighbor and exit through the side shared by the other excited neighbor; a path traced through a 3- or 4-neighbor C which has entered one side may exit through one of several sides. This ambiguity can be resolved by first identifying 2-neighbor C's crossed by the line outside the ambiguous region and then adjoining 3- and 4-neighbor C's, one at a time, that lie in a direction most likely to be on the line. For example, in Figure VI.9, trace the upper side to the right and adjoin an upper neighbor to a C if it is excited. If it is not, then adjoin the right neighbor. In case both excited right and upper neighbor occur, pick the upper neighbor and reject the right neighbor. This procedure

Figure VI.9: Interference of Excited Neighborhood Configurations Near Acute Vertices

assures that the uppermost path is taken through C's in the ambiguous region, and the direction (upward and to the right) is compatible with octant I slopes. For the lower side, adjoin the C to the right if it is excited and the C above otherwise, i.e., pick the lowest path through the ambiguous region.

Although this procedure is sequential, the number of C's in ambiguous regions is generally small unless the vertex angle is near 0^{o}. Such sharp angles are unusual in most polygon recognition contexts. Therefore, the path ambiguities can usually be resolved in a small number of steps. A more parallel method of resolving ambiguities based on the constraints of location and orientation of 3- and 4-neighbor C's on upper and lower sides has been examined. For example, note that a 3-neighbor C whose left neighbor is unexcited cannot occur on the upper side. Such C's can be immediately adjoined to the lower side no matter where they occur, along with their right and lower neighbors. The details of this process will be omitted as they are tediously long though straightforward.

A plane polygon contains a finite number of finite line segments, each of which shares its end points with exactly two others. For simplicity let there be no intersections other than end points. An analogous definition can be constructed for patterns to be recognized as polygons on a grid. Suppose a pattern of excited C's has been broken into chains of C's compatible with straight lines. The C's terminating these chains correspond to vertices, and each of them terminates two chains, but a "clump" of neighboring C's may have equal claims to be terminators.

Criteria for polygon recognition on the grid must admit ambiguitites due to the finite resolving power of the grid. The recognition algorithm must fit a set of straight line segments to the closed, connected sequence of cells representing the polygon by the approximation algorithm of VI.3, keeping only those with the smallest number of segments.

CHAPTER VII: CONCLUDING REMARKS

This study has investigated some pattern recognition capabilities of devices consisting of arrays of cooperating elements acting in parallel. The problem of recognizing straight lines in general position on the quadratic lattice has been completely solved by applying parallel acting algorithms to a special code for lines on the lattice. The experience of designing parallel computers embodying these algorithms led to the design of other parallel computers to recognize connectivity, detect boundaries, and approximate curves by straight line segments.

The use of cell synchronizers (CS's) did more than avoid synchronization problems. They were responsible for "executive decisions", and represent a higher level of hierarchical organization than the C's. Although a C and a CS are both finite state automata, their respective roles in the designs of recognizers here are quite different. Each C operates on inputs from its four nearest neighbors while the CS operates on inputs from a large collection of C's whose locations need not be given. In this sense the C's perform local computations and the CS performs "global" operations. This separation of roles resulted in considerable economy in logical complexity and computation time.

The saving in logical complexity occurs in the designs of the C's and CS. When all C's are to perform the same operation, their states need not be cluttered by what is common to all, it can be embodied in a single CS message to all C's. Similarly, a global property (such as

connectedness, linearity) to be recognized by a CS may be held by a large number of possible configurations of excited cells. If the CS were to account for the number and position of such configurations, it might need many more states than there are cells on the grid, since the set of sets sharing a property like connectedness is virtually a power set of the set of grid cells. This is one of the weaknesses of the perceptron which, with its fixed connections, must in some way account for all possible configurations of cells sharing the detected property.

The introduction of sequentiality from the alternation of C and CS operations is not drastic, for the C's operate in parallel. After each mass C operation the CS's determine what mass operation is performed next. This kind of joint operation yields greater flexibility in the capabilities of the array than purely parallel operation.

APPENDIX

ORIGIN OF THE CODE

Professor Jerome Rothstein

It is usual for a Ph.D. dissertation to rely heavily on the published papers of the candidate's advisor. In this case the relevant work was unpublished. I asked Mr. Weiman to include it in his dissertation as the text could not have been self-contained otherwise. He in turn asked me to write a capsule history of the genesis and development of the ideas so well exploited by him in constructing various cellular automata for pattern recognition and parallel computation. One of the reasons I agreed to do this is the instructive interplay between apparently unrelated fields the story brings out.

The straight line code had its origin in an entirely different context. Circa 1939-41, while a graduate student in physics, I sought a transparent example to clarify the points at issue in ergodic theory (from the viewpoint of elementary statistical mechanics). I hit on the idea of examining the trajectory of a specularly reflected perfect billiard ball in a square box, or what is the same thing, the path of a light ray in a square mirror box. Ergodicity shows up as a uniform distribution of reflection points along the box sides. I immediately realized that only lines with rational slopes gave periodic trajectories, so that non-ergodic situations had measure zero, that approximations of reals by rationals was important here, and that uncluttered diagrams could be drawn by treating the ray as a straight line on an infinite quadratic lattice whose cells were copies of the box. Successive

pieces of the reflected path were congruent respectively to successive segments lying within the cells on the plane. It was trivial to prove that length of path for a complete period depends only on the slope, and that the numbers of reflections by horizontal (H) and vertical (V) sides in a period was also invariant. Varying the initial point only permutes the string of H's and V's cyclically within the period. I realized that there was a one-to-one correspondence between rational numbers and a particular subset of the periodic strings over H and V, i.e., that I had a binary code for straight lines. To explore it further I was led to study some number theory, went through Hardy and Wright, perceived the relation of the subject to Farey series, continued fractions, and Minkowski's geometry of numbers, and saw how old the original problem and example were (Kronecker's theorem, p. 388 HW).

World War II led to other activities and a career in government and industry. The foregoing was virtually forgotten for almost thirty years, but was revived in an interesting way. My interest in statistical mechanics never abated, and the explosion of information theory since 1948 led me to consider informational generalization of physical entropy. It was natural to investigate "well-informed heat engines", their use in modeling biological systems, and slowly to shift into biophysics and into computer and information science. This culminated in 1967 in a joint appointment at the Ohio State University in both those departments. Exposure to open problems of visual systems, neural models, pattern recognition, formal languages, cellular automata, and parallel operations led to thinking about suitable simplified abstractions for dealing with

some of the crucial difficulties in bite-size pieces. This eventually narrowed down to the field giving this dissertation its title, and for reasons set forth in the introduction, to straight line recognition by a quadratic array of automata. The choice was probably subconciously influenced by the early development described above. In any event, it had a déjà-vu feel to it, jogged my memory, and I found odd sheets, scraps, and notebooks containing the old work, with all its student naïveté. In a very short time the basic code properties became clear to me, including the affine invariance property, the implied existence of simple recognition algorithms, and its suitability as a start in building logical languages adapted to cellular automata, pattern recognition, discretized geometry, and problems of parallel operation.

As Mr. Weiman began to develop from an assistant to a collaborator his contributions started to grow. To a first, and rather good, approximation, he designed the cellular automata to exploit the theory I developed. He did make some contributions to the theory however, like the results on negative continued fractions and the proof of code uniqueness given a shift pattern, and I made some to the cellular automata (choice of problem, topological background, some details, and the usual advisory and critical functions). I expect the field to become a fruitful one and look forward to many contributions by Mr. Weiman in the future.

BIBLIOGRAPHY

Amoroso, S., Lieblein, E. and Yamada, H. "A Unifying Framework for the Theory of Iterative Array Machines" Association for Computing Machinery Symposium on Theory of Computing, p. 259, 1969, ACM.

Block, H. D. "The Perceptron: a Model for Brain Functioning" Reviews of Modern Physics, Vol. 34, No. 1, pp. 123-135, 1962.

Burks, A. W. (Editor) Essays on Cellular Automata, University of Illinois Press, Chicago, 1970.

Caianiello, E. R. (Editor) Neural Networks (Proceedings of the School on Neural Networks, June 1967, in Ravello, Italy) Springer-Verlag, New York, 1968.

Codd, E. F. "Propagation, Computation, and Construction in Two-Dimensional Cellular Spaces", University of Michigan Ph.D. thesis, 1965. Published as book Cellular Automata, Academic Press, New York, 1968.

Duda, Richard O. "Some Current Techniques for Scene Analysis" Technical Report 46, Stanford Research Institute, Artificial Intelligence Group, October 1970.

Fu, K. S. Sequential Methods in Pattern Recognition and Machine Learning, Academic Press, New York, 1968.

Garner, H. L. and Squire, J. S. "Iterative Circuit Computers", in A. A. Barnum and M. A. Knapp (Eds.) Proceedings of a Workshop on Computer Organization, pp. 156-181, New York, Spartan Books, 1963.

Ginzburg, Abraham Algebraic Theory of Automata, New York, Academic Press, 1968, (ACM Monograph Series).

Gonzalez, R. "A Multi-Layer Iterative Circuit Computer", IEEE Transactions on Electronic Computers. EC-12, 5(1963), pp. 781-790.

Hardy, G. H. and Wright, E. M. An Introduction to the Theory of Numbers, (Fourth Edition) Oxford, Clarendon Press, 1965.

Holland, J. H. "A Universal Computer Capable of Executing an Arbitrary Number of Sub-programs Simultaneously," Proceedings of the Eastern Joint Computer Conference, Boston, pp. 108-113, 1959.

Holland, J. H. "Universal Embedding Spaces for Automata" in Norbert Wiener and J. P. Schade, (Eds.) Cybernetics of the Nervous System, (Progress in Brain Research 17), New York, Elsevier, pp. 223-243, 1965.

Khovanskii, Alexey Nikolaevitch The Application of Continued Fractions and Their Generalizations to Problems in Approximation Theory. (Translated by Peter Wynn) P. Noordhoff, N. V. Groningen, Netherlands, 1963.

McCulloch, W. S. and Pitts, W. H. "A Logical Calculus of the Ideas Immanent in Nervous Activity", Bulletin of Mathematical Biophysics, Vol. 9, pp. 127-247, 1943.

Minsky, Marvin and Papert, Seymour Perceptrons, M.I.T. Press, Cambridge, Mass., 1969.

Murtha, John C. "Highly Parallel Information Systems" in Franz L. Alt and Morris Rubinoff (Eds.), Advances in Computers V.7, London, Academic Press, p. 2, 1966.

Patrick, Edward A. Fundamentals of Pattern Recognition, Prentice-Hall, Inc., Englewood Cliffs, N.J., 1972.

Perron, O. Die Lehre von den Kettenbrüchen, Chelsea, New York, 1950.

Ratliff, F. Mach Bands: Quantitative Studies on Neural Networks in the Retina, Holden-Day, Inc., San Francisco, 1965.

Rektorys, Karel (Editor), Survey of Applicable Mathematics, M.I.T. Press, Cambridge, Mass., 1969.

Roberts, L. G. "Machine Perception of Threee-Dimensional Solid", in J. T. Tippett, et al., (Eds.) Optical and Electro-Optical Information Processing, M.I.T. Press, Cambridge, Mass., pp. 159-197, 1965.

Rosenblatt, F. Principles of Neurodynamics, Spartan Books, New York, 1962.

Sebestyen, G. S. Decision Making Processes in Pattern Recognition. The MacMillan Co., New York, 1962.

Sierpinski, W. Elementary Theory of Numbers, Panstwowe Wydawn. Naukowe (Polish Federal Scientific Publishers) Warsaw, 1964.

Von Neumann, J. (A. W. Burks, Ed.) Theory of Self-Reproducing Automata, University of Illinois Press, Urbana, Illinois, 1966.

Wall, H. S. Analytic Theory of Continued Fractions, D. Van Nostrand Co., New York, 1948.

Yamada, H. and Amoroso, S. "Tesselation Automata", Information and Control Vol. 14, pp. 299-317, 1969.

COMPUTER &
INFORMATION
SCIENCE
RESEARCH CENTER